MACHINE LEARNING FOR SPEECH FORENSICS AND HYPERSONIC VEHICLE APPLICATIONS

by

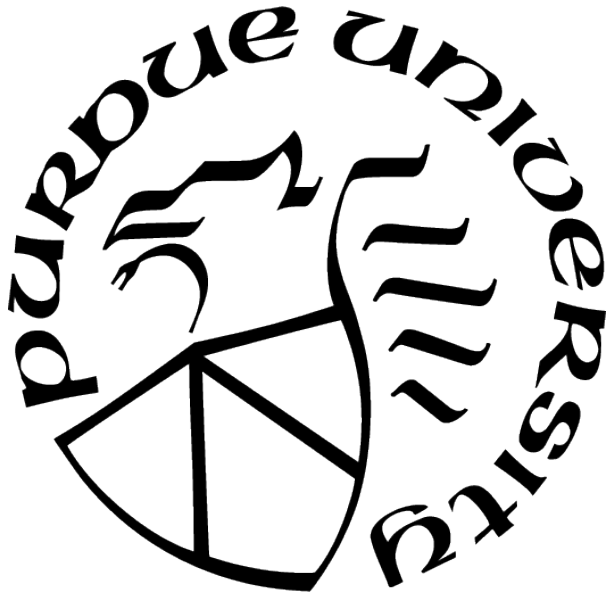
Emily R. Bartusiak

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy



School of Electrical and Computer Engineering

West Lafayette, Indiana

December 2022

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Edward J. Delp, Chair

School of Electrical and Computer Engineering

Dr. Amy R. Reibman

School of Electrical and Computer Engineering

Dr. Fengqing M. Zhu

School of Electrical and Computer Engineering

Dr. Mary L. Comer

School of Electrical and Computer Engineering

Approved by:

Dr. Dimitri Peroulis

For my love, David.

Thanks for supporting me in the challenging times and

celebrating with me in the good times.

ACKNOWLEDGMENTS

Thank you to my doctoral advisor, Professor Edward J. Delp, for his mentorship and friendship over the years. It has been a pleasure to work in the VIPER lab with him, and I am grateful for all of the guidance he has provided to me and the other VIPER Lab students. Thanks also to the members of my doctoral committee – Professor Amy R. Reibman, Professor Fengqing M. Zhu, and Professor Mary L. Comer – for their advice and feedback. I am grateful to have had them as role models throughout graduate school and beyond. I would like to express my gratitude to Professor Amy R. Reibman especially for her encouragement, advocacy, and friendship. I appreciate the conversations we have had throughout the years and the wisdom you have imparted.

Thank you to all of the former and current members of the VIPER Lab for their help, companionship, and inspiration. I treasure our talks in the VIPER Lab and over food that have challenged me and pushed me to be a better researcher. Thank you especially to Dr. David Güera, Dr. Sri Kalyan Yarlagadda, Dr. Daniel Mas, and Dr. Hanxing Hao for their close collaboration and mentorship.

I would like to especially thank the people who encouraged me to pursue a PhD in the first place: Dr. David Güera, Dr. Sri Kalyan Yarlagadda, Dr. Khalid Tahboub, Dr. Amy R. Reibman, and Dr. Charles A. Bouman. Thank you for helping me to visualize a different journey for myself than I thought possible and believing in me. I hope that one day I can provide the same encouragement to someone who needs it.

Finally, thank you to my family for their constant love, support, patience, and understanding throughout the completion of this dissertation.

The material in Chapters 2, 3, 4, and 5 is based on research sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under agreement number FA8750-20-2-1004. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, AFRL, or the U.S. Government.

The material in Chapters 6, 7, and 8 is based on research sponsored by Lockheed Martin Corporation. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied of Lockheed Martin Corporation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ADIOS	Automatic Distillation of Structure
AI	Artificial Intelligence
AFRL	Air Force Research Laboratory
AoA	Angle of Attack
AoAR	Angle of Attack Rate
ASV	Automatic Speaker Verification
AUC	Area Under the Curve
CAT	Compact Attribution Transformer
CCT	Compact Convolutional Transformer
CNN	Convolutional Neural Network
CQCC	Constant Q Cepstral Coefficient
CRV	Conic Re-entry Vehicle
DARPA	Defense Advanced Research Projects Agency
dB	Decibels
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DL	Deep Learning
DP	Dynamic Pressure
FFT	Fast Fourier Transform
FT	Fourier Transform
GAN	Generative Adversarial Network
GEP	Generalized Earley Parser
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HGV	Hypersonic Glide Vehicle
Hz	Hertz
KNN	K-Nearest Neighbors

LSTM	Long Short-Term Memory
MFCC	Mel Frequency Cepstral Coefficient
ML	Machine Learning
MLP	Multilayer Perceptron
NN	Neural Network
NLP	Natural Language Processing
PaSST	Patchout faSt Spectrogram Transformer
PCFG	Probabilistic Context-Free Grammar
PG	Parse Graph
PR	Precision Recall
RF	Random Forest
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
SemaFor	Semantic Foresnsics
SS	Speech Synthesis
SSAST	Self-Supervised Audio Spectrogram Transformer
SVM	Support Vector Machine
TALO	Time After Lift-Off
tSNE	t-Distributed Stochastic Neighbor Embedding
VC	Voice Conversion

ABSTRACT

Synthesized speech may be used for nefarious purposes, such as fraud, spoofing, and misinformation campaigns. We present several speech forensics methods based on deep learning to protect against such attacks. First, we use a convolutional neural network (CNN) and transformers to detect synthesized speech. Then, we investigate closed set and open set speech synthesizer attribution. We use a transformer to attribute a speech signal to its source (*i.e.*, to identify the speech synthesizer that created it). Additionally, we show that our approach separates different known and unknown speech synthesizers in its latent space, even though it has not seen any of the unknown speech synthesizers during training. Next, we explore machine learning for an objective in the aerospace domain.

Compared to conventional ballistic vehicles and cruise vehicles, hypersonic glide vehicles (HGVs) exhibit unprecedented abilities. They travel faster than Mach 5 and maneuver to evade defense systems and hinder prediction of their final destinations. We investigate machine learning for identifying different HGVs and a conic reentry vehicle (CRV) based on their aerodynamic state estimates. We also propose a HGV flight phase prediction method. Inspired by natural language processing (NLP), we model flight phases as "words" and HGV trajectories as "sentences." Next, we learn a "grammar" from the HGV trajectories that describes their flight phase transition patterns. Given "words" from the initial part of a HGV trajectory and the "grammar", we predict future "words" in the "sentence" (i.e., future HGV flight phases in the trajectory). We demonstrate that this approach successfully predicts future flight phases for HGV trajectories, especially in scenarios with limited training data. We also show that it can be used in a transfer learning scenario to predict flight phases of HGV trajectories that exhibit new maneuvers and behaviors never seen before during training.

1. INTRODUCTION

Terms such as "artificial intelligence" (AI) and "machine learning" (ML) evoke imagery of futuristic scenes with omniscient robots and hovercrafts. Although neither robots nor hovercrafts are prevalent in today's society, the pursuit of intelligent machines enchants humankind. The scientific community strives to develop and harness the power of machines that perform advanced computational methods in a way that imitates human intelligence. Furthermore, the community investigates the application of such machine intelligence to various domains, including speech recognition, visual object recognition, object detection, drug discovery, and genomics [1]. The history of machine learning dates back to the 1950s in the early days of pattern recognition [2], [3], when Samuel first coined the term [4]. Today, machine learning research is more popular than ever before [5].

Breakthroughs in concepts such as backpropogation [6]–[9] and hardware resources [10]–[12] enabled machine learning methods to learn from more data at scale. Moreover, novel architectures succeeding at difficult tasks demonstrated the potential of machine learning [13]–[17]. These methods are often considered deep learning (DL) methods because they consist of many computational layers that operate sequentially on an input, creating a deeper network. Such networks are referred to as neural networks (NNs). As computational resources and algorithms continue to increase in efficiency and processing power, the number of computational layers can also increase, adding even more depth to NNs. Because of these advancements and promising results, research trends in machine learning, deep learning, and artificial intelligence have increased exponentially in recent years [5]. Figure 1.1 shows the dramatic increase in publications from zero in 1986 to over 25,000 in 2016 using the term "deep learning."

Machine learning methods draw conclusions from data in an induction approach [18]. In other words, ML methods analyze data and discover trends directly from the data itself. Through an extensive and iterative analysis of a large amount of data, ML methods improve over time as they "learn" the most salient features of a dataset necessary to make certain decisions. For this reason, it is important that the dataset supplied to a ML approach for training (*i.e.*, sampled distribution, training dataset) is representative of the total distri-

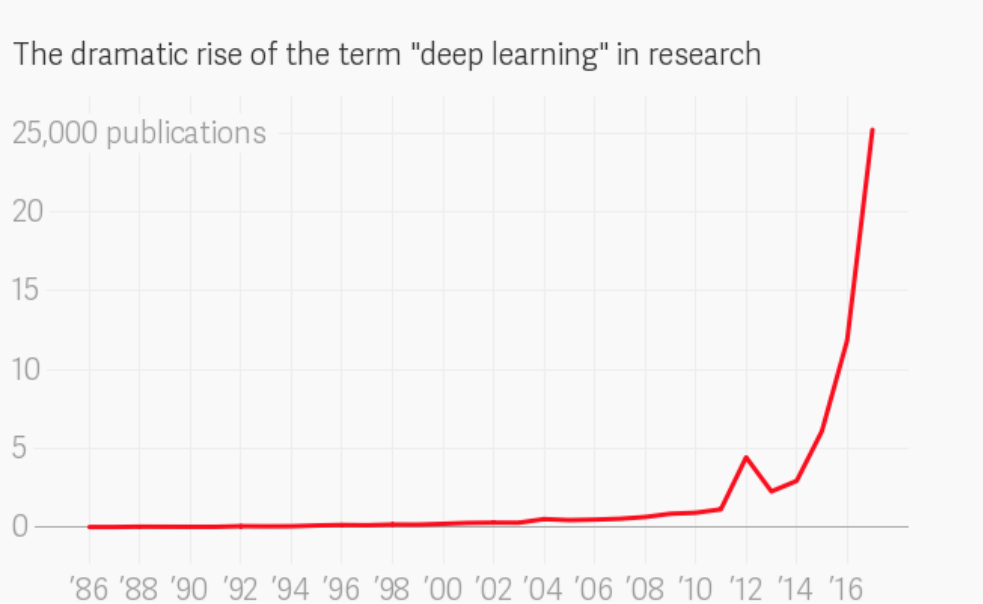


Figure 1.1. Prevalence of Deep Learning Research. This graph depicts the exponentially increasing trend in recent years of research publications that use the term "deep learning." Source: [5].

bution that a ML method may evaluate in a real-world scenario (*i.e.*, population, testing dataset). The dataset features provided to ML algorithms should be grounded in mathematical, engineering, scientific, and physics-based reasoning.

In this dissertation, machine learning and deep learning are used to analyze data from two different domains: speech forensics and hypersonics. In both domains, we begin our investigation with supervised learning approaches in scenarios in which the training data is representative of the testing data. For speech forensics, this entails development of synthesized speech detection methods that distinguish between synthesized speech signals and speech signals containing authentic, recorded human voices. For hypersonics, this entails vehicle classification to identify the type of vehicle flying. After demonstrating success on these tasks, we advance to more complicated scenarios and explore unsupervised learning on out-of-distribution tasks. In other words, we investigate machine learning methods in scenarios in which the training data is not necessarily representative of the testing data. For speech forensics, this entails detecting synthesized speech on a sequestered dataset and developing a speech synthesizer attribution method that can discriminate between different

speech synthesizers in an open set scenario. For hypersonics, this entails predicting future behavior and engageability of HGVs and using transfer learning to apply a model trained on one dataset containing a certain maneuver to another dataset that exhibits a new maneuver. The application of machine learning to problems in these domains started with hypotheses grounded in reality and evolved through investigation and analysis to the results presented today. The rest of this chapter will provide introductions to these specific problems, contributions of this work, and publications resulting from the investigation.

1.1 Speech Forensics

Nowadays, it is common to interact with synthesized speech on a daily basis. We talk with Siri, Alexa, and Google Assistant in our homes, on our phones, and in our cars [19]–[23]. Virtual assistants answer customer service phone numbers and help us with anything from scheduling appointments to paying our bills to accessing bank accounts [24]–[26]. Often, we can easily tell that the voices of these virtual assistants are synthesized from their robotic tones. In some situations, though, it is more difficult to discern whether we are listening to synthesized or authentic human voices.

Social media platforms offer many tools that allow users to create new speech signals that sound realistic. Both Tiktok and Instagram provide text-to-speech (TTS) services, enabling users to generate new speech signals with custom messages [27]–[30]. The platforms maintain libraries of voice styles, any of which can be used to deliver the messages. The libraries include standard human voices as well as fun characters, such as C-3P0 and a Stormtrooper [31]–[33]. Tiktok also offers an option for users to upload an audio track of a specific voice style they would like to replicate. If users already have an audio track with a desired message, they can transform the message into a different voice style using Tiktok. Deep learning methods for speech synthesis and voice conversion systems can also generate realistic-sounding human speech [34]–[37]. Because many easy-to-use tools like these exist for modifying speech with high quality, the quantity of inauthentic speech is increasing rapidly [38]. Although all of these features can be used for comedic purposes, they can also easily be used with more detrimental consequences.

Attackers may generate new speech signals impersonating people for a variety of malevolent ambitions. For example, they could create a new message for a public figure with the hope that the speech goes viral and disseminates misinformation. When inauthentic video accompanies inauthentic speech signals, such as in deepfakes, the potential to influence public opinion and current events is even higher [39], [40]. In 2022, a deepfake arose that showed Ukrainian President Volodymyr Zelensky surrendering to Russia [41], [42]. Although it was quickly debunked, this scenario shows the potential a realistic deepfake with inauthentic audio can have on war situations and other global matters. In another example, Goldman Sachs stopped a $40 million investment in 2021 after realizing they were conducting business with an impersonator using synthesized speech on a conference call [43]. As people conduct more and more business remotely, they must authenticate their virtual interactions more than ever [44]–[46]. In this dissertation, we propose several speech forensics methods to counter attacks like these.

In Chapter 2, we present a method that analyzes speech signals to determine whether they contain genuine, recorded human voices or synthesized human voices (*i.e.*, voices generated by neural acoustic and waveform models). Instead of analyzing the speech signals directly, our proposed approach converts the speech signals into spectrograms displaying frequency, intensity, and temporal information and evaluates them with a CNN. Trained on both genuine speech signals and synthesized speech signals, we show that our approach achieves high accuracy on this classification task.

In Chapter 3, we continue our investigation of synthesized speech detection. Again, we analyze the speech signals in the form of spectrograms. This time, though, use a compact convolutional transformer (CCT) to analyze them. We demonstrate that this approach improves synthesized speech detection over our prior CNN work. We also show that this approach succeeds on a sequestered dataset as part of a DARPA competition included in the Semantic Forensics (SemaFor) program [47].

In Chapter 4, we investigate three transformers to detect synthesized speech: compact convolutional transformer (CCT), patchout fast spectrogram transformer (PaSST), and self-supervised audio spectrogram transformer (SSAST). We show that each transformer independently detects synthesized speech well. Then, we propose an ensemble of transformers

that can provide even better performance. Finally, we explore how much of a speech signal is needed for high synthesized speech detection. In this approach, we are still analyzing audio in the frequency domain, rather than analyzing audio waveforms directly. However, now we explore speech signals represented as mel spectrograms, which show frequencies of the speech signals according to the mel scale [48]. The mel scale is a perceptual scale based on the human auditory system [48]. We demonstrate that our transformer ensemble achieves the highest synthesized speech detection results of all methods explored in this dissertation. Furthermore, we show that it can detect synthesized speech with the same level of high success when analyzing shorter amounts of a speech signal.

In Chapter 5, we present our final speech forensics investigation. So far, we have focused on synthesized speech detection methods. These methods are important for protection against audio-based fraud, spoofing, and misinformation attacks. Speech attribution methods provide even more information about the nature of synthesized speech signals because they identify the specific speech synthesis method (*i.e.*, speech synthesizer) used to create a speech signal. Due to the increasing number of realistic-sounding speech synthesizers, we propose a speech attribution method that generalizes to new synthesizers not seen during training. To do so, we investigate speech synthesizer attribution in both a closed set scenario and an open set scenario. In other words, we consider some speech synthesizers to be "known" synthesizers (*i.e.*, part of the closed set) and others to be "unknown" synthesizers (*i.e.*, part of the open set). We represent speech signals as spectrograms and train our proposed method, known as compact attribution transformer (CAT), on the closed set for multi-class classification. Then, we extend our analysis to the open set to attribute synthesized speech signals to both known and unknown synthesizers. We utilize a t-distributed stochastic neighbor embedding (tSNE) on the latent space of the trained CAT to differentiate between each unknown synthesizer. Additionally, we explore poly-1 loss formulations to improve attribution results. Our proposed approach successfully attributes synthesized speech signals to their respective speech synthesizers in both closed and open set scenarios.

1.2 Hypersonic Vehicle Applications

One result of scientific progress is the ability to perform tasks more quickly and efficiently. Sometimes, the need for speed is the motivating factor behind a research endeavor. Nowadays, we expect our desires to be met quickly and accurately. Many products and services emerged with the goal of providing immediate satisfaction. Text and email communications, finding answers to questions, ordering products, requesting taxis, and streaming services are all examples of tasks that consumers expect to complete instantaneously as soon as the idea pops into their heads. As society moves faster and faster, we desire our vehicles to move faster and more autonomously as well. For example, traveling across the continental United States in the past from New York to San Francisco via horse and buggy would take four to six months and require constant navigation supervision [49]. However, flying across the U.S. along this same path would require only seven hours today without the need to navigate personally [50]. In the future, this trip could take only two hours, thanks to hypersonic vehicles [51], [52].

Hypersonic vehicles fly faster than Mach 5 (*i.e.*, five times the speed of sound). In order to fly at such tremendous speeds, some hypersonic vehicles utilize a jet engine called a supersonic combustion ramjet, often referred to as a scramjet [53]. Other hypersonic vehicles use a rocket engine [52]. Spacecrafts reach hypersonic speeds as they re-enter the atmosphere of a planet, yet typically they do not have any kind of engine [54]. In addition to high speed, high maneuverability differentiates hypersonic vehicles from other types of vehicles.

Hypersonic vehicles exhibit a high degree of agility, which manifests itself in two main ways. First, hypersonic vehicles possess high turning ability, which allows them to perform different maneuvers during flight. For example, hypersonic vehicles may swerve and dive around obstacles. Second, hypersonic vehicles may disguise their final destinations. The flight paths of hypersonic vehicles are challenging to predict due to the vehicle's agility. Vehicles may land in any number of locations within a large coverage area, depending on the final maneuver executed to land a vehicle. As a result, the actual destination can be ambiguous until the final descent [55].

Due to the development of new hypersonic vehicles, we need methods to recognize and characterize them, as we do with other vehicles. It is important to distinguish hypersonic vehicles from other types of vehicles as well as discriminate between different types of hypersonic vehicles. In Chapter 6, we endeavor to classify both hypersonic vehicles and non-hypersonic vehicles. We present multiple machine learning methods that analyze aerodynamic features of vehicles over time as they fly along their trajectories. We utilize a k-nearest neighbors (KNN) classifier, support vector machine (SVM), and convolutional neural network (CNN) to determine whether a trajectory under analysis belongs to one type of hypersonic glide vehicle *(HGV-1)*, another type of hypersonic glide vehicle *(HGV-2)*, or a non-hypersonic vehicle called a conic re-entry vehicle *(CRV)*.

HGVs fly at very high velocities and demonstrate high agility, which creates challenges in forecasting their flight behavior (*e.g.*, aerodynamics, flight paths, maneuvers). In Chapter 7, we describe a method for predicting future flight phases of a HGV. Flight phases could be modeled as categorical labels that correspond to different types of flight behavior. For our purposes, we define flight phases based on magnitude of rate of change of total energy (*i.e.*, summation of kinetic and potential energy) of a vehicle. We use methods from natural language processing (NLP) to model the flight phases as "words" and the HGV trajectories as "sentences." We learn a "grammar" from the HGV trajectories, which we use for our prediction task. Given "words" from the initial part of a HGV trajectory and the "grammar", we can predict future "words" in the "sentence" (*i.e.*, future HGV flight phases in the trajectory). We demonstrate that this approach successfully predicts future flight phases for HGV trajectories, especially in scenarios with limited training data.

There are limited examples of actual HGV flights, which impedes the development of prediction methods that model HGV trajectories. In Chapter 8, we investigate transfer learning for HGV trajectory prediction to evaluate how stochastic grammars trained on a limited number of HGV trajectories perform on new, unseen HGV trajectories. Our analysis includes two datasets containing HGV trajectories that exhibit different maneuvers. One dataset contains trajectories that exhibit vertical maneuvers, which are behaviors related to changes in altitude. The second dataset exhibits both horizontal and vertical maneuvers, where horizontal maneuvers refer to changes in crossrange and downrange. The vertical

dataset is also significantly smaller than the second dataset (*i.e.*, the dataset that exhibits horizontal and vertical maneuvers). We develop a prediction method using the smaller, less complicated dataset to model HGV trajectories. Specifically, we use an unsupervised machine learning method based on stochastic grammars. Then, we demonstrate that the learned grammar can be used to predict HGV behavior and engageability for trajectories from the larger, more complicated dataset. Our results show that transfer learning improves prediction performance (even on unseen trajectory maneuvers) and can be used in limited data scenarios.

1.3 Contributions of this Dissertation

The contributions of this dissertation are as follows:

- we investigate a CNN and multiple transformers for synthesized speech detection and demonstrate success on this task;
- we show that our transformer-based synthesized speech detection approach also succeeds on a sequestered dataset of lossy audio signals, which indicates that it is able to generalize to new, unseen data samples not present in the training dataset;
- we design a transformer ensemble to improve synthesized speech detection;
- we demonstrate that our transformer ensemble can achieve high success on this task, even when analyzing shorter portions (*i.e.*, less) of a speech signal;
- we propose a compact attribution transformer (CAT) to attribute synthesized speech signals to known and unknown speech synthesizers;
- we investigate the latent space of CAT with tSNE to distinguish between different speech synthesizers and offer more insight into the trained neural network;
- we demonstrate that CAT successfully discriminates between different known and unknown synthesizers, which indicates that CAT generalizes to new speech synthesizers not seen during training;

- we use poly-1 loss formulations to improve attribution results;
- we investigate a CNN for vehicle classification based on initial phases of a vehicle's flight;
- we demonstrate that machine learning approaches successfully discriminate between three types of vehicles (two different hypersonic vehicles and one conic re-entry vehicle), even under noisy conditions;
- we introduce stochastic grammars to model HGV trajectories;
- we demonstrate that our stochastic grammar method accurately predicts HGV flight phases, even with a limited amount of training data;
- we show that our vehicle classification and HGV prediction methods achieve higher success as time after lift-off (TALO) increases;
- we propose an engageability definition that indicates when a HGV could be intercepted;
- we use transfer learning to derive a stochastic grammar with a smaller dataset of HGV trajectories that exhibit vertical maneuvers and predict future behavior and engageability of HGV trajectories in a larger dataset that exhibit both horizontal and vertical maneuvers;
- we demonstrate that transfer learning with stochastic grammars can predict HGV behavior and engageability, even when the method encounters HGV trajectories that exhibit different maneuvers than the trajectories used to derive the stochastic grammar.

1.4 Publications

Journal Publications Resulting from this Dissertation:

1. **E. R. Bartusiak**, M. A. Jacobs, M. W. Chan, M. L. Comer, and E. J. Delp, "Predicting Hypersonic Glide Vehicle Behavior with Stochastic Grammars", in *IEEE Transactions on Aerospace and Electronics*, 2023, *Submitted for Review.*

Book Chapters Resulting from this Dissertation:

2. H. Hao*, **E. R. Bartusiak***, D. Güera, D. Mas, S. Baireddy, Z. Xiang, S. K. Yarlagadda, R. Shao, J. Horváth, J. Yang, F. Zhu, and E. J. Delp, "Deepfake Detection Using Multiple Data Modalities", in *Handbook of Digital Face Manipulation and Detection - From DeepFakes to Morphing Attacks, Series on Advances in Computer Vision and Pattern Recognition*, Springer, Cham, January 2022, pp. 235-254. DOI: 10.1007/978-3-030-87664-7_11.

Conference Publications Resulting from this Dissertation:

3. **E. R. Bartusiak**, K. Bhagtani, A. K. S. Yadav, and E. J. Delp, "Transformer Ensemble for Synthesized Speech Detection", 2023, *Submitted for Review.*

4. **E. R. Bartusiak**, M. A. Jacobs, C. F. Spells, M. W. Chan, M. L. Comer, and E. J. Delp, "Transfer Learning for Hypersonic Vehicle Trajectory Prediction", *Proceedings of the IEEE Aerospace Conference*, pp. 1-8, March 2023, Big Sky, MT, USA.

5. **E. R. Bartusiak** and E. J. Delp, "Transformer-Based Speech Synthesizer Attribution in an Open Set Scenario", *Proceedings of the IEEE International Conference on Machine Learning and Applications*, pp. 1-8, December 2022, Nassau, The Bahamas. DOI: 10.48550/arXiv.2210.07546.

*denotes equal authorship.

6. **E. R. Bartusiak**, H. Hao, M. A. Jacobs, N. X. Nguyen, M. W. Chan, M. L. Comer, and E. J. Delp, "A Stochastic Grammar Approach to Predict Flight Phases of a Hypersonic Glide Vehicle", *Proceedings of the IEEE Aerospace Conference*, pp. 1-15, March 2022, Big Sky, MT, USA. DOI: 10.1109/AERO53065.2022.9843362.

7. **E. R. Bartusiak** and E. J. Delp, "Synthesized Speech Detection Using Convolutional Transformer-Based Spectrogram Analysis", *Proceedings of the IEEE Asilomar Conference on Signals, Systems, and Computers*, pp. 1426-1430, October 2021, Asilomar, CA, USA (Virtual). DOI: 10.1109/IEEECONF53345.2021.9723142.

8. **E. R. Bartusiak**, N. X. Nguyen, M. W. Chan, M. L. Comer, and E. J. Delp, "A Machine Learning Approach to Classify Hypersonic Vehicle Trajectories", *Proceedings of the IEEE Aerospace Conference*, pp. 1-14, March 2021, Big Sky, MT, USA (Virtual). DOI: 10.1109/AERO50100.2021.9438274.

9. **E. R. Bartusiak** and E. J. Delp, "Frequency Domain-Based Detection of Generated Audio", *Proceedings of the Media Watermarking, Security, and Forensics Conference, IS&T Electronic Imaging Symposium*, pp. 273(1)-273(7), January 2021, Burlingame, CA, USA (Virtual). DOI: 10.48550/arXiv.2205.01806.

Other Publications:

10. K. Bhagtani, **E. R. Bartusiak**, A. K. S. Yadav, P. Bestagini, and E. J. Delp, "Synthesized Speech Attribution Using the Patchout Spectrogram Attribtion Transformer", 2023, *Submitted for Review.*

11. A. K. S. Yadav, Z. Xiang, **E. R. Bartusiak**, P. Bestagini, S. Tubaro, and E. J. Delp, "ASSD: Synthetic Speech Detection in the AAC Compressed Domain", 2023, *Submitted for Review.*

12. A. K. S. Yadav, **E. R. Bartusiak**, K. Bhagtani, and E. J. Delp, "Synthetic Speech Attribution Using Self Supervised Audio Spectrogram Transformer", *Proceedings of the*

Media Watermarking, Security, and Forensics Conference, IS&T Electronic Imaging Symposium, January 2023, San Francisco, CA, USA.

13. **E. R. Bartusiak**, M. Barrabés, A. Rymbekova, J. Gimbernat-Mayol, C. Lopéz, L. Barberis, D. Mas, X. Giró-i-Nieto, and A. G. Ioannidis, "Predicting Dog Phenotypes from Genotypes", *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 3558-3562, July 2022, Glasgow, Scotland, UK. DOI: 10.1109/EMBC48229.2022.9870905.

14. K. Bhagtani, A. K. S. Yadav, **E. R. Bartusiak**, Z. Xiang, R. Shao, S. Baireddy, and E. J. Delp, "An Overview of Recent Work in Multimedia Forensics: Methods and Threats", pp. 1-17, November 2022, *arXiv:2204.12067*. DOI: 10.48550/arXiv.2204.12067.

15. K. Bhagtani, A. K. S. Yadav, **E. R. Bartusiak**, Z. Xiang, R. Shao, S. Baireddy, and E. J. Delp, "An Overview of Recent Work in Multimedia Forensics", *Proceedings of the IEEE International Conference on Multimedia Information Processing and Retrieval*, pp. 324-329, August 2022, San Jose, CA, USA (Virtual). DOI: 10.1109/MIPR54900.2022.00064.

16. E. D. Cannas, S. Baireddy, **E. R. Bartusiak**, S. K. Yarlagadda, D. Mas, P. Bestagini, S. Tubaro, and E. J. Delp, "Open-Set Source Attribution for Panchromatic Satellite Imagery", *Proceedings of the IEEE International Conference on Image Processing*, pp. 3038-3042, September 2021, Anchorage, AK, USA (Virtual). DOI: 10.1109/ICIP42928.2021.9506600.

17. H. Hao, S. Baireddy, **E. R. Bartusiak**, L. Konz, K. LaTourette, M. Gribbons, M. Chan, M. Comer, and E. J. Delp, "An Attention-Based System for Damage Assessment Using Satellite Imagery", *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, pp. 4396-4399, July 2021, Brussels, Belgium (Virtual). DOI: 10.1109/IGARSS47720.2021.9554054.

18. H. Hao, S. Baireddy, **E. R. Bartusiak**, M. Gupta, K. LaTourette, L. Konz, M. Gribbons, M. Chan, M. Comer, and E. J. Delp, "Building Height Estimation via Satellite

Metadata and Shadow Instance Detection", *Proceedings of the SPIE Automatic Target Recognition Conference*, pp. 1-16, April 2021, Orlando, FL, USA (Virtual). DOI: 10.1117/12.2585012.

19. D. Mas, H. Hao, S. K. Yarlagadda, S. Baireddy, R. Shao, J. Horváth, J. Yang, **E. R. Bartusiak**, D. Güera, F. Zhu, and E. J. Delp, "Deepfakes Detection with Automatic Face Weighting", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Workshop on Media Forensics*, pp. 2851-2859, June 2020, Seattle, WA, USA (Virtual). DOI: 10.1109/CVPRW50498.2020.00342.

20. **E. R. Bartusiak**, "An Adversarial Approach to Spliced Forgery Detection and Localization in Satellite Imagery", *Purdue University*, pp. 1-54, May 2019, West Lafayette, IN, USA. DOI: 10.25394/PGS.8035952.v1.

21. **E. R. Bartusiak**, S. K. Yarlagadda, D. Güera, F. Zhu, P. Bestagini, S. Tubaro, and E. J. Delp, "Splicing Detection And Localization In Satellite Imagery Using Conditional GANs", *Proceedings of the IEEE International Conference on Multimedia Information Processing and Retrieval* , pp. 91-96, March 2019, San Jose, CA, USA. DOI: 10.1109/MIPR.2019.00024.

2. SYNTHESIZED SPEECH DETECTION WITH A CNN

2.1 Overview

Synthesized media can be generated for multiple different modalities, including text, images, videos, and audio. Technological advancements enable people to generate manipulated or false multimedia content relatively easily. Because generating false content is so accessible, the quantity of synthesized media increases exponentially daily [38]. Fabricated content has been used for years for entertainment purposes, such as in movies or comedic segments. However, it also has the potential to be introduced for nefarious purposes. Audio authentication is necessary for speaker verification. If audio is synthesized to impersonate someone successfully, an adversary may access personal devices with confidential information, such as banking details and medical records. Furthermore, fabricated audio could be used in the audio tracks of deepfake videos.

In this chapter, we consider an audio authentication task. The reason for this is twofold. First, there are cases in which the only medium available is audio, such as in a speaker verification task. Second, there are cases in which multiple types of data are available for analysis, such as a deepfake detection task, which would benefit from a multi-modal analysis that includes fake audio detection. Our method examines audio signals in the frequency domain in the form of spectrograms, as shown in Figure 2.1.

A spectrogram is a visualization technique for audio signals. It shows the relationship between time, frequency, and intensity (or "loudness") of an audio signal – all in the same graph. Time increases from left to right along the horizontal axis, while frequency increases from bottom to top along the vertical axis. Colors densely fill the middle of the graph and indicate the strength of a signal over time at different frequencies. Much like a heat map, brighter colors depict greater strength. We treat these spectrograms as images and analyze them using deep learning techniques to determine whether an audio track is *genuine* or *synthesized*.

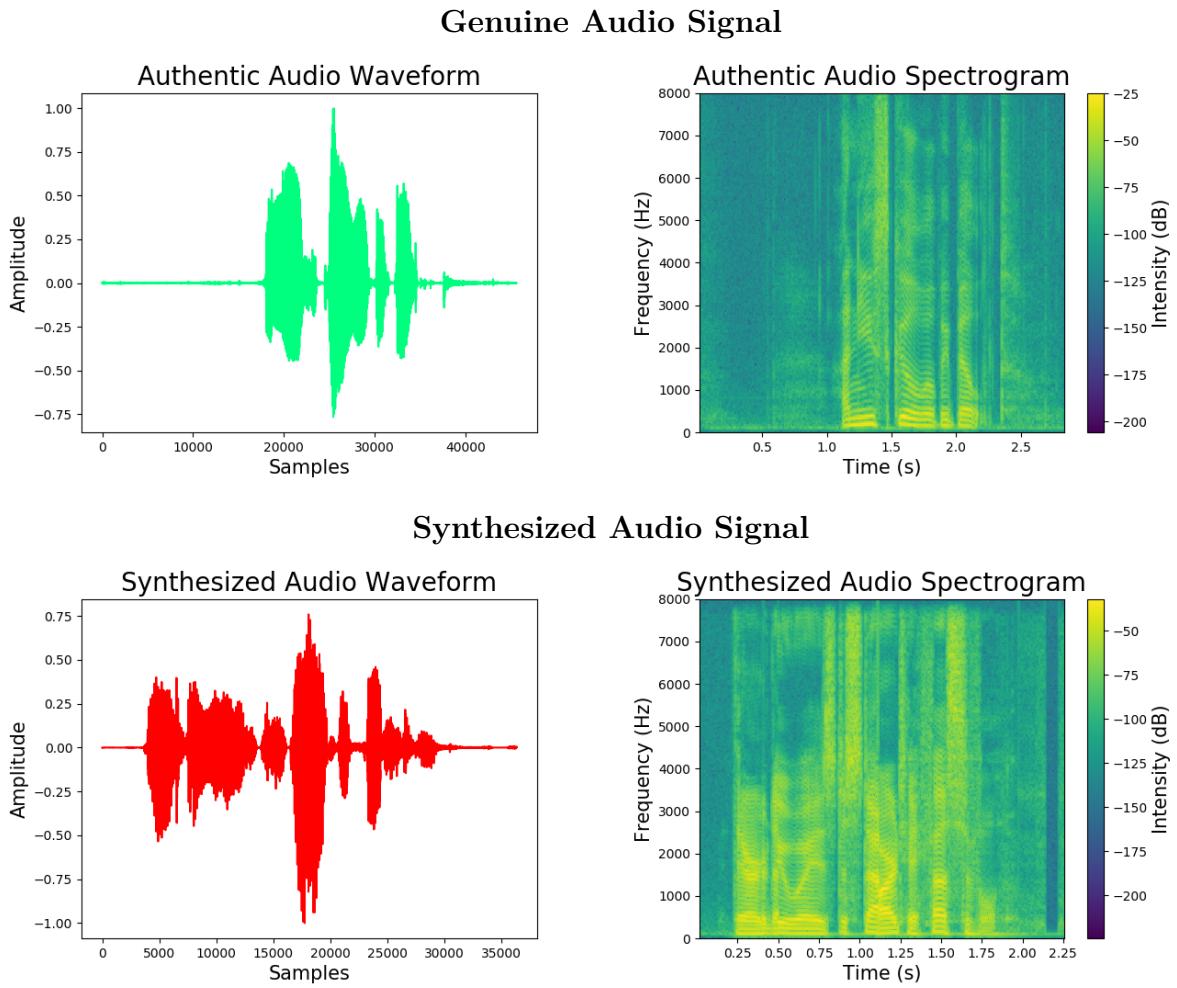


Figure 2.1. Audio Signals and Spectrograms. *Left column:* Audio signals in the time domain (*i.e.*, waveforms), where the top image shows a genuine audio signal spoken by a human and the bottom image shows a synthesized audio signal. *Right column:* Spectrograms generated from the time domain audio signals, which are used by the CNN to determine audio authenticity.

2.2 Related Work

Developing methods to verify multimedia is an ongoing research effort. Previous work includes analysis of audio content [56], visual content [57] [39] [58] [59], metadata [60], and combinations of these modalities [61].

2.2.1 Spoofing Attacks

For the audio modality specifically, spoofing attacks fall into three main categories: voice conversion (VC), speech synthesis (SS), and replay attacks. Voice conversion refers to the process of transforming an already existing speech signal into the style of another speaker, so that it sounds as if a new speaker is saying exactly what the original speaker said. Speech synthesis refers to methods in which new audio signals are generated from scratch. For example, converting written text into spoken speech (*i.e.*, text-to-speech (TTS)) is one method to achieve speech synthesis. Finally, replay attacks refer to spoofing methods in which the original speaker and desired speech are recorded. Then, this speech signal is played back to an audio-capturing device, which is fooled into believing the replayed audio signal is the desired speaker in real-time. Some research efforts, such as [56], focus on replay attacks specifically. On the other hand, we focus solely on voice conversion and speech synthesis attacks, which consist of synthetically generated audio signals.

2.2.2 Audio Features

Digital signal processing offers many different methods to extract features to analyze audio signals. Arguably the most famous method for signal analysis is the Fourier Transform (FT) and its subsidiaries (*e.g.*, Discrete Fourier Transform (DFT)), which deconstruct a function of time into its constituent frequencies. Many techniques build upon the foundation of the Fourier Transform. Constant Q Cepstral Coefficients (CQCCs) are derived by converting a signal from time domain to frequency domain with the Fourier Transform, spacing the spectral amplitudes logarithmically, and then converting the amplitudes to the quefrency domain with a time scale [62]. Mel Frequency Cepstral Coefficients (MFCCs) are also based on the Fourier Transform. In order to compute MFCCs, the Fourier Transform is applied to time domain audio signals, and the powers of the resulting spectrum are mapped onto the mel scale [48]. The mel scale describes how humans perceive tones and pitches. It reflects humans' sensitivity to different frequencies [63]. Next, the logarithmic scale is applied to the powers at each of the mel frequencies in preparation to compute the Discrete

Cosine Transform (DCT) [64] of the mel log powers. Finally, the amplitudes of the result of the DCT constitute the MFCCs [65].

Besides enabling the computation of feature coefficients, the Fourier Transform may be used to construct visual representations of signals. Nowadays, spectrogram generation is a digital process which involves sampling a signal in the time domain, dividing the signal into smaller segments, and applying the Fourier Transform to each segment to calculate the magnitude of the frequency spectrum for each segment. Through this process, a segment corresponding to a specific moment in time is transformed into a sequence of spectrum magnitude values. To construct a graph of these values, the sequence is oriented as a vertical line and color coded according to magnitude, creating a vertical line of "pixels." These pixel lines are concatenated side-by-side in order of increasing time index to construct a spectrogram "image."

2.2.3 Audio Authentication Approaches

Current audio authentication methods utilize the aforementioned features to determine whether an audio signal is real or fake. They first estimate the desired audio features from the time domain signal (*i.e.*, waveform) and then use them as inputs to a deep learning system. For example, [66] uses CQCCs and MFCCs as inputs to a standard multilayer perceptron network (MLP) and ResNet-based CNN. Chen *et al.* investigate CQCCs and MFCCs as inputs to long short-term memory networks (LSTMs), gated recurrent unit networks (GRUs), and recurrent neural networks (RNNs) [56]. For these methods, the audio signals are represented as sequences of coefficients, which are then fed into a neural network. Conversely, some work analyzes audio signals directly. In such cases, the method relies on the learning-based system to identify relevant audio features. Chintha *et al.* use audio signals as inputs to a CNN-LSTM model, where the first few layers of the network consist of convolution layers and a later layer consists of a LSTM layer [61]. The authors also explore working with log-melspectrograms, which are spectrograms in which the frequency domain content is mapped to the mel scale. The log-melspectrograms are analyzed with a CNN to detect authentic and spoofed audio.

Independent from audio authentication tasks, many signal processing research efforts use spectrograms for a variety of other human speech-related tasks. Verma *et al.* explore a style transfer method for audio signals which transforms a reference signal into the style of a specific target signal [67]. This work utilizes both audio waveforms and spectrograms as inputs to a CNN architecture. Dennis *et al.* classify sound events based on spectrograms with a SVM [68]. Jeng *et al.* investigate audio signal reconstruction based on spectrograms [69]. Works such as [70] endeavor to improve upon the traditional spectrogram and focus on underlying, stable structures grounded in the lower frequencies of an audio signal. More recently, there have been efforts to analyze spectrograms with respect to emotions. Stolar *et al.* use a CNN to analyze spectrograms and differentiate between seven different emotions captured in speakers' voices [71]. Zheng *et al.* analyze spectrograms with a CNN and then feed the extracted CNN features into a random forest (RF) to identify speakers' emotions [72]. Prasomphan uses a MLP to analyze spectrograms for the purpose of detecting emotion of audio signals [73], [74]. Mittal *et al.* and Malik explore an emotion recognition task and fake audio detection task in tandem [75], [76]. He *et al.* use a gaussian mixture model (GMM) and KNN to detect stress in speech signals [77]. Inspired by these works conducted for more general tasks in the signal processing domain, we leverage a CNN that analyzes spectrograms.

Our approach takes advantage of the translation invariant properties of images to find critical, local indicators revealing the authenticity of an audio signal. Furthermore, our approach benefits from shared weights which collectively learn from all patches of a spectrogram. By leveraging signal processing techniques, image processing techniques, and deep learning techniques, we detect authentic and inauthentic audio clips with high reliability and accuracy.

2.3 Proposed Approach

We investigate an audio discrimination task in this chapter. Given an audio signal of a few seconds in length, we seek to recognize whether it is genuine human speech or synthesized speech. Our overall approach is shown in Figure 2.2.

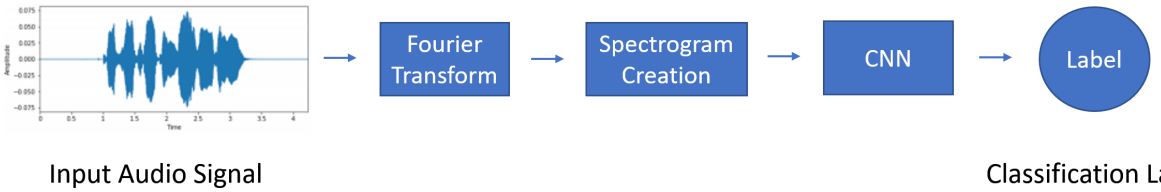


Figure 2.2. Proposed Synthesized Speech Detection Method. The proposed approach transforms audio signals in the time domain into spectrograms, which are used as inputs to a CNN. Then, the CNN produces a classification label indicating whether the signal under analysis is *authentic* or *synthesized*.

2.3.1 Spectrogram Creation

The first step in our analysis is to consider the digital audio signal in the time domain. Let $f(t)$ be a continuous time domain audio signal where t is the time index. Figure 2.1 and Figure 2.3 show examples of time domain audio signals (*i.e.*, waveforms). By a visual inspection, it is unclear which signals could be genuine and which could be synthesized. In order to leverage computer vision techniques for forensic analysis, we convert these time domain signals into frequency domain spectrograms, as shown Figure 2.1 and Figure 2.3.

The conversion process involves taking the Discrete Fourier Transform (DFT) of a sampled signal $f[n]$ to obtain Fourier coefficients $F(m)$, where m is the frequency index in hertz (Hz). The magnitudes of the coefficients $|F|$ are then color coded to indicate the strength of the signal. $f[n]$ refers to a sampled, discrete version of $f(t)$ with a total of N samples. The N samples can be denoted as $f[0], f[1], ..., f[N-1]$, where each sample $f[n]$ is an impulse with area $f[n]$. The DFT is:

$$F(m) = \sum_{n=0}^{N-1} f[n] e^{-\frac{i2\pi}{N} mn} \tag{2.1}$$

A Fast Fourier Transform (FFT) is a method that efficiently computes the DFT of a sequence. Therefore, we utilize the FFT to rapidly obtain Fourier coefficients $F(m)$ of the signals in our dataset. For our experiments, we run the FFT on blocks of the signal consisting of 512 sampled points with 511 points of overlap between consecutive blocks. The

signals in our dataset have a sample rate of 16 kHz, so the audio signals are sliced into equally-sized temporal segments of 32 milliseconds in length. Once the Fourier coefficients have been computed, the audio signal $f[n]$ is converted to decibels for magnitude scaling: $f_{dB} = 10\log(|f|)$. The spectrogram "image" of size 50x34 pixels is then constructed to show the audio signal's magnitude (*i.e.*, intensity in dB) over time versus frequency, as shown in Figure 2.3.

Each spectrogram encompasses information from an entire audio track. We can determine frequencies and intensities of an audio signal as it propagates in time by analyzing the colors in the spectrogram from left to right in the image. The warmer and more yellow a color

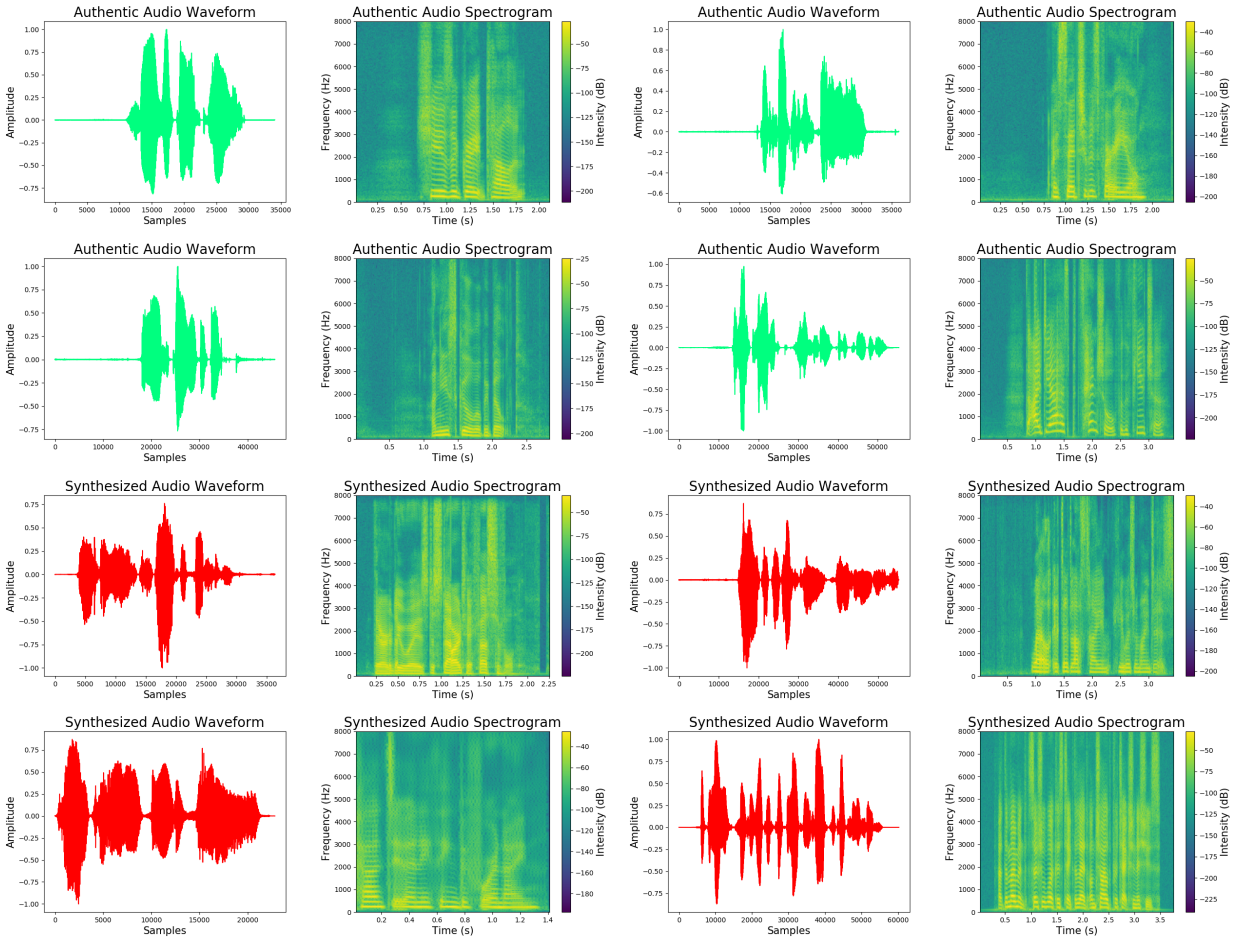


Figure 2.3. Audio Waveforms and Spectrograms. Genuine and synthesized audio signals analyzed by the CNN.

is, the louder the audio signal is at that point in time and at that frequency. Darker colors indicate quieter sounds. Once the spectrogram images are created, they are converted to grayscale images and normalized in preparation for analysis by the CNN.

2.3.2 Convolutional Neural Network (CNN)

We employ a convolutional neural network (CNN) to analyze the normalized, grayscale spectrogram images and detect whether they represent genuine or synthesized audio. Table 2.1 outlines the specifics of the network architecture depicted in Figure 2.4. It consists mainly of two convolutional layers in the initial stages of the CNN. Then, it employs max pooling and dropout for regularization purposes and to prevent overfitting. The final output of the neural network applies a softmax function to a fully-connected dense layer of two nodes, producing two final detection scores. The scores indicate the probabilities that the audio segment under analysis is considered to be *genuine* or *synthesized*. Finally, the argmax function is used to convert these probabilities into a final class prediction. We train for 10 epochs using the Adam optimizer [78] and cross entropy loss function.

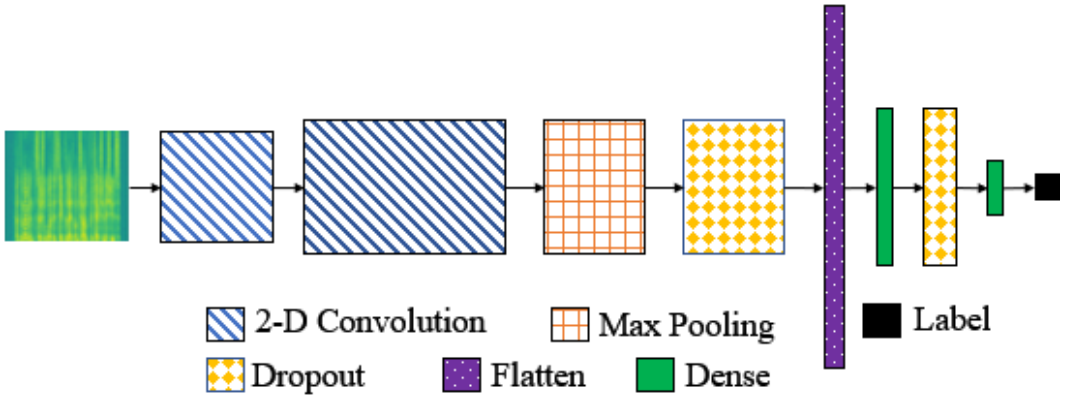


Figure 2.4. CNN Diagram for Synthesized Speech Detection. The CNN developed for our audio authentication approach.

Table 2.1. CNN Architecture for Synthesized Speech Detection. This table indicates the parameters of the proposed CNN. Each row in the table specifies *(from left to right)* the details of the layer, its output shape, and the number of parameters it contains. Output shape is in the form (N, H, W), where N refers to the number of feature maps produced, H refers to the height of the feature maps produced, and W refers to the width of the feature maps produced.

CNN Architecture		
Layer	Output Shape (N, H, W)	Parameters
$conv_1$	(32, 48, 32)	320
$conv_2$	(30, 46, 64)	18,496
max pooling	(15, 23, 64)	0
$dropout_1$	(15, 23, 64)	0
$flatten_1$	(22080)	0
$dense_1$	(128)	2,826,368
$dropout_2$	(128)	0
$dense_2$	(2)	258

2.4 Experimental Setup

2.4.1 Dataset

To validate our methods, we utilize the ASVspoof2019 dataset [79]. This dataset was introduced in the *ASVspoof2019: Automatic Speaker Verification Spoofing and Countermeasures Challenge* [80]. It contains both genuine human speech samples and fabricated speech samples. The inauthentic speech samples fall into the three categories outlined in Chapter 2.1, Section 2.2.1: voice conversion (VC), speech synthesis (SS), and replay attacks. For this work, we only consider generated audio. Thus, we only utilize the VC and SS subsets of the dataset. The synthesized audio was generated with neural acoustic models and deep learning methods, including LSTMs [81] and generative adversarial networks (GANs) [82]. Our final version of the dataset based on only VC and SS attacks contains 121,461 audio tracks. The details of the dataset are included in Table 2.2. We utilize the official dataset split according to the challenge, which results in 25,380 training tracks, 24,844 validation

tracks, and 71,237 testing tracks. The average length of all of the audio signals in the entire dataset (including training, validation, and testing samples) is 3.35 seconds.

Table 2.2. ASVspoof2019 Dataset. Details about the dataset used in our experiments.

ASVspoof2019 Dataset					
Subset	Genuine Speech Signals	Synthesized Speech Signals	Total Speech Signals	Female Speakers	Male Speakers
Training	2,580	22,800	25,380	12	8
Validation	2,548	22,296	24,844	6	4
Testing	7,355	63,882	71,237	27	21
Total	12,483	108,978	121,461	45	33

2.4.2 Experimental Results

Table 2.3 summarizes the results of our method. For comparison purposes, we evaluate how our approach performs relative to a baseline approach in which the classifier randomly guesses whether an audio signal is *genuine* or *synthesized* according to a uniform random distribution. Our spectrogram-CNN achieves 85.99% accuracy on the testing dataset, outperforming the baseline random method by 35.95%. This indicates that our method is considerably better than random chance. The precision, recall, and F1-scores of our method are 67.23%, 75.94%, and 70.08%, respectively. These values further indicate that even on an unbalanced dataset, our method performs well.

Table 2.3. Results for Synthesized Speech Detection. This table indicates the performances of the baseline random method and our proposed method.

Synthesized Speech Detection Results				
Method	Accuracy	Precision	Recall	F1
Baseline (Random)	50.06%	49.93%	49.80%	40.63%
Proposed Method	**85.99%**	**67.23%**	**75.93%**	**70.08%**

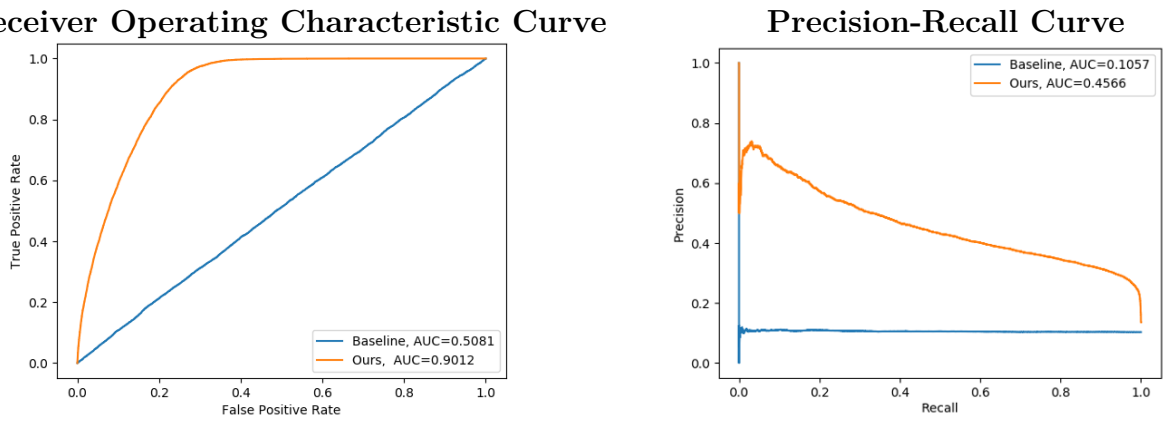


Figure 2.5. ROC and PR Curves. Our method is in orange (the top line in both plots), and the baseline random approach is in blue (the bottom line in both plots).

Figure 2.5 shows the Receiver Operating Characteristic (ROC) and Precision-Recall (PR) curves of our method in comparison to the baseline random method. For both of these plots, the ideal, completely accurate classifier would yield curves resembling a 45-degree angle that include the top-left corners of the plots. The closer a curve to that corner, the better a classifier performs. A way to measure the quality of a curve is by calculating the area under the curve (AUC). A higher AUC value indicates a better classifier, with an AUC of 1.0 indicating a "perfect" classifier. Because PR AUC does not depend on the class distribution of the dataset, it is a useful metric for evaluating classifiers on unbalanced datasets such as ours. Our method yields a high ROC AUC of 0.9012 and a PR AUC of 0.4566. In comparison to the baseline method which achieves a ROC AUC of 0.5081 and a PR AUC of 0.1057, our method performs better by both metrics.

Considering that the testing dataset contains new audio attacks (*i.e.*, new speech synthesis methods) which were never seen before in training, these results are very promising. They demonstrate that our method generalizes well to some unseen audio attacks. However, there are still some other unseen attacks on which our method fails, and more investigation into its failure cases is needed. In general, though, a CNN analysis of audio signals formatted as spectrograms is effective for an audio verification task.

3. SYNTHESIZED SPEECH DETECTION WITH A TRANSFORMER

3.1 Overview

In this chapter, we continue our work for synthesized speech detection but pivot to explore a different type of network. Now, we use a compact convolutional transformer (CCT) to analyze speech signals in the form of spectrograms. CCT utilizes a convolutional layer that introduces inductive biases and shared weights into a network, allowing a transformer architecture to perform well when fewer data samples are available for training. CCT uses an attention mechanism to incorporate information from all parts of a signal under analysis. Trained on both genuine human voice signals and synthesized human voice signals, we demonstrate that our CCT approach successfully differentiates between genuine and synthesized speech signals. We also show that this approach succeeds on a sequestered dataset as part of a DARPA competition included in the Semantic Forensics (SemaFor) program [47].

3.2 Related Work

Although prior deep learning work relies heavily on CNNs, recent developments indicate that convolutions may not be necessary to effectively analyze images [83], [84]. These methods succeed in image classification tasks without inductive biases provided by convolutions. Inspired by the success of attention mechanisms in natural language processing (NLP) [85], vision transformer (ViT) analyzes patches of an image with an attention mechanism for image classification tasks [83]. Hassani *et al.* adopt concepts from CNNs and ViT to create a compact convolutional transformer (CCT) [86]. CCT leverages the inductive biases and efficiencies of parameter-sharing that convolutions provide to succeed at machine learning tasks with smaller-sized datasets compared to the datasets used with ViT. It also leverages the attention mechanism of transformers to capture long-range dependencies in images. CCT combines the power of convolutions with the power of transformers. We utilize a CCT trained on spectrograms showing genuine and synthesized speech signals to identify synthesized speech.

3.3 Proposed Approach

Figure 3.1 shows an overview of our approach, known as compact convolutional transformer (CCT). CCT uses a standard transformer encoder, as used in [83], [85]. However, CCT introduces two new features – a convolutional image encoding block and a sequence pooling layer – that replace operations in standard transformer approaches. The CCT first uses a convolutional block (*i.e.*, a series of convolutional layers) to embed an input image into a latent space. In our experiments, we utilize two convolutional layers with a kernel of size 3x3, a ReLU activation function, and max pooling. The first and second convolutional layers produce sets of 64 and 128 feature maps, respectively. We use this convolutional block instead of the standard transformer practice of dividing input images into non-overlapping patches, which contain only local image information and fail to preserve information at patch boundaries. The feature maps contain aggregate information from all regions of an image, so they are more salient inputs to the transformer encoder. Because they result from convolution operations, they also introduce inductive biases to the network. This enables the transformer to train more efficiently, which is highly important on smaller-sized datasets. Next, we row concatenate each of the 128 2D feature maps (sized 32x32) into a vector of length 1024, creating the tokens analyzed by the transformer encoder. We use positional embedding (a standard practice in transformers) for each token so that the transformer understands how they relate spatially [83], [87].

Next, CCT analyzes the tokens with the transformer encoder, which consists of two transformer encoder layers. Each transformer encoder layer contains the multi-headed attention mechanism that captures long-range dependencies between different parts of the input. The transformer encoder layers are modeled after typical attention-based layers [83], [85]. Then, sequence pooling occurs on the outputs of the transformer encoder. The pooling operation smooths the sequence of outputs so that the MLP Head can correctly detect whether the speech signal under analysis is *synthesized* or *genuine*. Sequence pooling also eliminates the need for an extra token (*i.e.*, a classification token) that other transformers use [83], [87]. With sequence pooling, the model no longer needs to track the classification token throughout its layers.

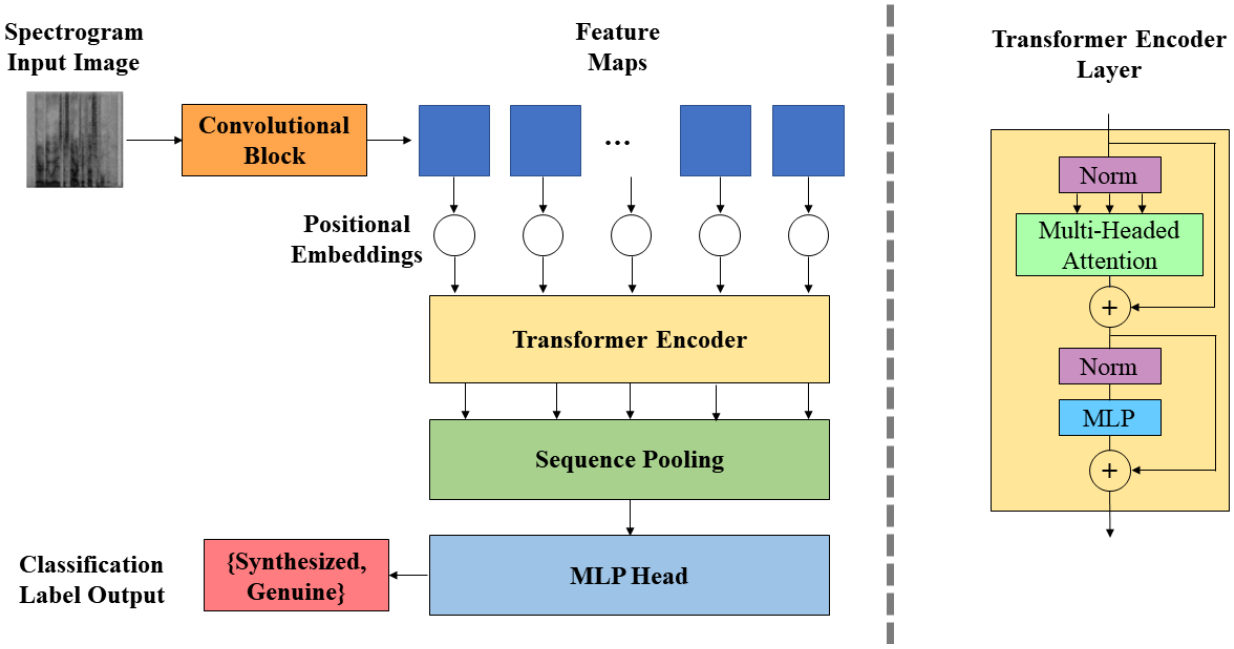


Figure 3.1. Block diagram of Our Synthesized Speech Detection Approach based on a Compact Convolutional Transformer. This block diagram shows a spectrogram of size 128x128 being analyzed by a compact convolutional transformer (CCT), which produces a classification output indicating if the spectrogram represents genuine or synthesized speech.

3.4 Experimental Setup

3.4.1 Dataset and Spectrogram Creation

We again utilize the ASVspoof2019 dataset [79] in our experiments. Table 3.1 summarizes the details of the dataset. Recall that the ASVspoof2019 dataset is heavily imbalanced, with significantly more synthesized speech signals than genuine speech signals.

We convert speech waveforms from the dataset into spectrograms by following a similar procedure as described in [88]. More specifically, we use the Fast Fourier Transform (FFT) to compute Fourier coefficients of signals in our dataset. The FFT operates on blocks of the signals consisting of 512 sampled points with 128 points of overlap between consecutive blocks. Then, the Fourier coefficients are converted to decibels and organized in 2D arrays to construct the spectrograms. We represent each spectrogram with a matrix of 128x128 values. Note that these spectrograms are larger than those used in our previous approach in

Table 3.1. ASVspoof2019 Dataset. Details about the dataset used in our experiments.

ASVspoof2019 Dataset					
Subset	Genuine Speech Signals	Synthesized Speech Signals	Total Speech Signals	Female Speakers	Male Speakers
Training	2,580	22,800	25,380	12	8
Validation	2,548	22,296	24,844	6	4
Testing	7,355	63,882	71,237	27	21
Total	12,483	108,978	121,461	45	33

Chapter 2. The larger spectrograms have higher resolution, which preserves more details of speech signals for the synthesized speech detector. Next, we perform min-max normalization on the intensity values, mapping the spectrogram intensities to the range of values [0,1]. Normalized values enable machine learning models to learn more quickly because they are forced to focus on relative rather than absolute differences in input values. Figure 3.2 shows an example of a grayscale, normalized spectrogram that is analyzed by the CCT.

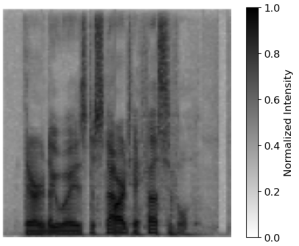


Figure 3.2. Spectrogram of a Synthesized Speech Signal. Our CCT approach analyzes grayscale, normalized spectrograms of size 128x128.

3.4.2 Comparison Methods

To validate our approach, we compare it against several other methods. First, we establish three baseline methods: Baseline-Minority, Baseline-Majority, and Baseline-Prior. Baseline-Minority is a classifier that only predicts that speech signals belong to the minority class – in this case, the *genuine* category. Baseline-Majority is a classifier that does the opposite. It only predicts that speech signals belong to the *synthesized* class. Considering

the significant class imbalance in the dataset, we expect Baseline-Minority to have the worst performance of all methods. Meanwhile, Baseline-Majority establishes a threshold above which a classifier actually performs well. Baseline-Prior is the final baseline classifier. It randomly assigns a label to a signal under analysis according to the known distribution of *genuine* vs. *synthesized* samples in the training data split. In addition to these baselines, we investigate the effectiveness of KNN [89], [90], SVM [91], and logistic regression (LogReg) [92] on this task. These methods operate on row concatenated versions of the 128x128-sized spectrograms (*i.e.*, vectors of length 16,384). Finally, we compare our results to our previous synthesized speech detection method from Chapter 2 that utilizes a CNN. We report the performance of the CNN on the smaller-sized spectrograms (dimensions 50x34 pixels) as well as the the new, larger-sized spectrograms (dimensions 128x128).

3.4.3 Experimental Results on Public Dataset

Table 3.2 and Figure 3.3 show the results of all approaches. We report accuracy, weighted precision, weighted recall, weighted F1, Balanced Accuracy, Receiver Operating Characteristic Area Under the Curve (ROC AUC), and Precision Recall Area Under the Curve (PR AUC) [93], [94]. Weighted metrics are computed with a weighted average of each metric obtained on the two classes, where weights reflect the dataset class imbalance. Results indicate that the CCT approach outperforms all other methods by a clear margin. It achieves the highest metrics of all methods considered. Although the CCT performs better than our previously proposed CNN, the CNN achieves the second highest performance when trained on spectrograms of size 128x128 pixels. Overall, the two neural network approaches perform the best.

Results confirm that both larger input spectrograms and the new model contribute to better performance. Comparing the results of CNN-50x34 and CNN-128x128, we observe that balanced accuracy, ROC AUC, and PR AUC increase when the CNN is trained and evaluated on larger, higher-resolution inputs. In this case, the CNN is presented with more detailed input images that allow it to better discriminate genuine and synthesized speech signals. Comparing the results of CNN-128x128 and CCT, we observe that ROC AUC and

Table 3.2. Results of All Methods for Synthesized Speech Detection. ROC AUC and PR AUC represent area under the curve of the receiver operating characteristic and precision recall curves, respectively.

Synthesized Speech Detection Results							
Method	Accuracy	Weighted Precision	Weighted Recall	Weighted F1	Balanced Accuracy	ROC AUC	PR AUC
Baseline-Minority	10.32%	1.07%	10.32%	1.93%	50.00%	0.5000	0.1032
Baseline-Prior	81.59%	81.46%	81.59%	81.52%	49.94%	0.4994	0.1032
Baseline-Majority	89.68%	80.42%	89.68%	84.79%	50.00%	0.5000	0.1032
KNN	89.45%	84.99%	89.45%	85.57%	52.08%	0.6751	0.2643
LogReg	80.60%	91.76%	80.60%	83.98%	84.41%	0.9041	0.4101
CNN-50x34	85.59%	90.50%	85.59%	87.35%	79.26%	0.9052	0.4649
SVM	89.93%	90.94%	89.93%	85.25%	50.47%	0.9113	0.5023
CNN-128x128	85.27%	93.21%	85.27%	87.60%	**89.22%**	0.9416	0.6278
CCT	**92.13%**	**93.79%**	**92.13%**	**92.70%**	87.78%	**0.9646**	**0.7501**

PR AUC increase even more when an attention mechanism is used. The attention mechanism of the transformer determines the most important part of a spectrogram and focuses on that part of the image more so than the less discriminative regions, which aids in its detection capabilities. However, transformers have historically required very large-scale datasets in order to learn properly, suffering from a lack of inductive biases that CNNs have. Because we utilize convolutional layers, the CCT achieves a greater degree of shared weights, learns more efficiently, and leverages the inductive biases to achieve high success, even with fewer data samples from which to learn.

3.4.4 Experimental Results on Sequestered Dataset

Now that we have demonstrated successful synthetic audio detection on a publicly available dataset, we investigate the performance of our proposed approach on a sequestered, private dataset. The dataset is part of a competition in the Semantic Forensics (SemaFor) program organized by DARPA [47]. The SemaFor program focuses on methods that detect, attribute, and characterize synthesized and manipulated multimedia. The SemaFor evalu-

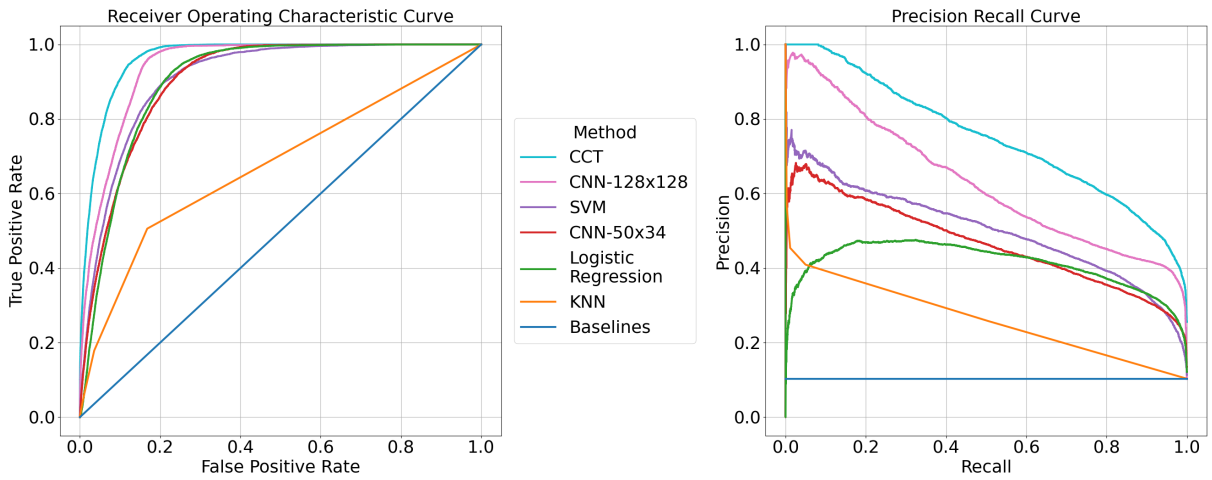


Figure 3.3. ROC and PR Curves of All Methods for Synthesized Speech Detection. The three baseline methods are all represented by the "baselines" method. Our CCT approach outperforms all other methods.

ation team organizes multiple competitions related to falsified media, such as a deepfake detection, synthesized speech detection, generated text detection, text and image inconsistencies detection, and attribution verification related to authors of news articles. In all of these competitions, the evaluation data is sequestered. In other words, competition participants do not have direct access to the evaluation data to use in development of their media forensics methods. Instead, participants must develop their methods in a way that generalizes to new, unseen data from an unknown distribution.

We submitted our transformer-based synthesized speech detection method to a SemaFor competition related to generated audio detection. This competition evaluates methods' abilities to detect both synthesized and manipulated audio. Synthesized audio refers to audio signals that have been completed generated by an audio generation method. Manipulated audio refers to audio signals in which only some segments have been generated, while other segments are authentic, recorded audio clips. In other words, manipulated audio signals are authentic audio signals in which synthesized audio has been spliced into them at one or more temporal locations. We do not have more details about the audio signals in the dataset. For example, we do not know which speech synthesis methods were used to create the speech signals; we do not know how many different speakers are included in the authentic

audio signals; and we do not know the sources of the audio signals (*e.g.*, YouTube, internet websites, social media, television). The audio signals could be from a pristine, noiseless recording environment, or they could be heavily and/or multiply compressed (in the case where they have been shared multiple times online).

At the time of submission, our transformer-based synthesized speech detection method achieved top-3 performance in this competition out of more than twenty-five different submissions. Figure 3.4 shows the ROC curves of the five methods that achieve the highest generated audio detection results in this competition. Our proposed approach (shown in green), achieves a ROC AUC of 0.7705. Although this value is lower than the ROC AUC score achieved on ASVspoof2019 (which was 0.9646), a ROC AUC value 0.7705 is still fairly good, especially considering that we have no prior knowledge of the characteristics exhibited in the evaluation data or access to it for training our method.

There are a few differences between the training and evaluation scenarios that could contribute to this drop in ROC AUC. First, the evaluation data in the SemaFor competition presents a new challenge to our method in the form of manipulated/spliced audio signals. Our method is trained on audio signals that are either entirely authentic or entirely synthesized. In the SemaFor competition, some audio signals may be a combination of both, though. Since our method has never seen audio signals like these, it is reasonable that our approach may experience a drop in detection. Second, the evaluation data could be significantly longer than audio signals used to train our approach. Our training dataset consists of audio signals that are 3.35 seconds on average. In the SemaFor competition, there are some audio signals that are several minutes long. To analyze new audio signals, our approach uses the initial portion of an audio signal to create a spectrogram with 128 temporal windows. This spectrogram size captures most (if not all) of an audio signal that is only a few seconds long. However, it does not capture all of an audio signal that lasts for several minutes. Thus, our current approach only uses the initial portion of an audio signal to determine whether it is synthesized. Because the audio signals in the SemaFor competition are longer and are constructed with splicing, it is possible that the initial portion of an audio signal analyzed by our method is entirely authentic but that a spliced portion of synthesized audio appears later in the audio signal. Our approach would not detect the synthesized audio, in this

case. One final point to consider is the difference in quality between training and evaluation audio signals. The ASVspoof2019 dataset stores audio signals as FLAC files, which use a lossless audio compression codec. Thus, we train our synthesized speech detection method on high-quality audio signals, which may make it easier to expose synthesized speech signals. By comparison, all audio signals in the SemaFor generated audio competition are stored as MP3 files, which use a lossy audio compression codec. This means that the evaluation data does not preserve as much information as is present in the training data. Despite this loss in quality, our trasnformer-based synthesized speech detection method is able to extend to lossy audio files. Our SemaFor competition results indicate that our proposed approach generalizes to new audio signals of an unknown nature, even if they are longer, lossy, and spliced.

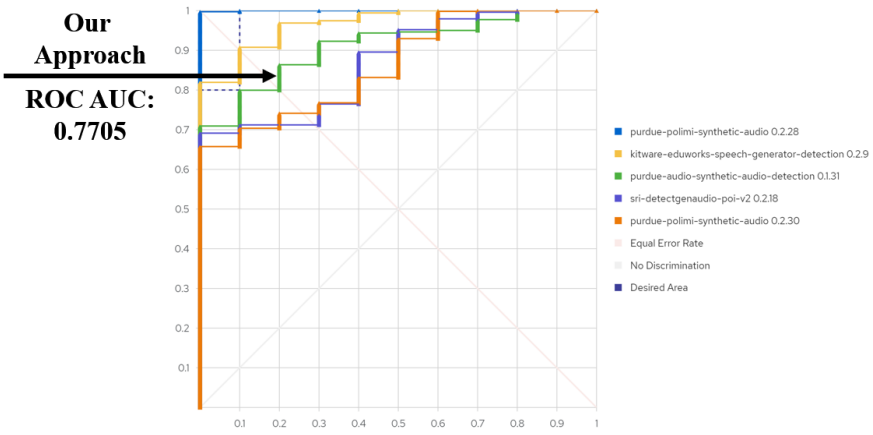


Figure 3.4. ROC Curves of Top-5 Performers on Sequestered Dataset. This plot shows ROC curves from the top-5 methods for generated audio detection in a SemaFor competition. Our transformer-based synthesized speech detector is shown in green and achieves top-3 performance.

4. SYNTHESIZED SPEECH DETECTION WITH A TRANSFORMER ENSEMBLE

4.1 Overview

In this chapter, we investigate three transformers for synthesized speech detection: compact convolutional transformer (CCT) [86]; patchout fast spectrogram transformer (PaSST) [95]; and self-supervised audio spectrogram transformer (SSAST) [96]. Our investigation focuses on transformers, which have achieved success on a variety of tasks in recent years. We explore whether we can harness the potential of transformers, which typically require large datasets to train, on a smaller, imbalanced dataset for synthesized speech detection. We train each of the transformers on mel spectrograms [48] to authenticate speech signals. Mel spectrograms are 2-D visual representations of audio signals showing frequency and intensity of an audio signal over time [48]. The frequencies of the audio signals are in the mel scale, which is a perceptual scale based on the human auditory system [48]. Although each of the transformers detects synthesized speech from mel spectrograms well on its own, we demonstrate that fusing the transformers in an ensemble achieves even better detection. Finally, we explore how much of an audio signal is needed for synthesized speech detection. We show that individual transformers are more sensitive to different lengths of audio signals. However, our transformer ensemble consistently and successfully detects synthesized speech from all mel spectrogram sizes (corresponding to different audio signal lengths) considered.

4.2 Related Work

There are many different ways to represent an audio signal: as a temporal waveform, as a spectrogram, or as sequences of coefficients from a transform (*e.g.*, Constant Q Cepstral Coefficients (CQCCs) [62]). Hua *et al.* analyze audio waveforms with a CNN based on ResNet and Inception networks [97]. However, waveform-based methods can struggle with longer audio signals because the inputs can be hundreds of thousands of samples. One way to create inputs of shorter lengths is to use sequences of transform coefficients of the audio signals. In these approaches, the audio waveforms are converted to sequences of

coefficients, such as CQCCs [62], and then analyzed. Chen *et al.* use a MLP, ResNet-based CNN, LSTM, GRU, and RNN to detect synthesized speech represented as CQCCs [56], [66]. Other methods represent audio signals as 2-D arrays that contain information about an audio signal's frequencies and intensity over time. These 2-D arrays are known as spectrograms and can be treated as images. Our prior work introduced in Chapters 2 and 3 analyze spectrograms of audio signals for synthesized speech detection using a CNN and a convolution transformer [88], [98], [99]. Spectrograms can represent frequencies according to the mel scale, which is a scale that represents pitches perceived to be equally distant according to the human auditory system [48]. These versions are known as mel spectrograms. Conti *et al.* analyze mel spectrograms to identify emotions and then use a random forest classifier on the emotion features to detect synthesized speech [100]. Gong *et al.* and Koutini *et al.* use mel spectrograms to classify over 600 different types of audio signals, from bird noises to specific speech commands [95], [96], [101]. Based on the success of work with mel spectrograms, we use mel spectrograms for synthesized speech detection.

4.3 Proposed Approach

4.3.1 Mel Spectrogram Creation

We convert speech waveforms into mel spectrograms by following a similar procedure as described in [95], [96], [101]. More specifically, we create mel spectrograms with 128 mel frequency bins. The mel spectrogram is computed using a 25 ms Hanning window with a shift of 10 ms. In other words, an audio waveform is divided into 25 ms "time frames" or "windows" with 10 ms overlap between consecutive windows. We orient the mel spectrograms so that the height of the mel spectrograms is 128 (corresponding to the 128 mel frequency bins) and the width of the spectrograms corresponds to the length of the audio signal (in terms of 25 ms time frames). We crop or zero-pad the mel spectrograms to each of the following dimensions: 128x256, 128x512, and 128x1024. Notice that only the second dimension (indicating the temporal length of the audio signal) is affected by the cropping and padding procedure. The first dimension (corresponding to the number of mel frequency bins) always remains the same. Figure 4.1 shows mel spectrograms of the same audio signal

formatted as different sizes (*i.e.*, formatted to analyze different lengths of an audio signal). These mel spectrograms are used to train and evaluate the transformers and transformer ensemble.

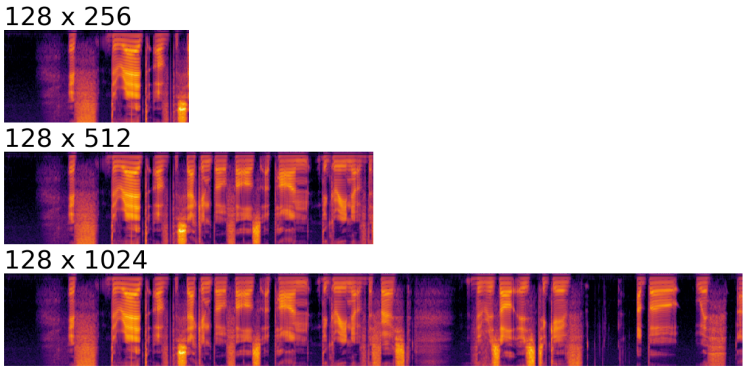


Figure 4.1. Mel Spectrograms Showing the Same Speech Signal Cropped to Different Lengths. Different lengths correspond to different amounts of time of the speech signal. The first dimension (*i.e.*, height of mel spectrogram) is always 128, corresponding to 128 mel frequency bins. The second dimension (*i.e.*, width/length of mel spectrogram) indicates the number of temporal windows of an audio signal used in the analysis.

4.3.2 Compact Convolutional Transformer (CCT)

Compact convolutional transformer (CCT) [86] is a smaller transformer used for analyses on 2-D inputs, such as our mel spectrogram inputs. It uses a series of convolutional layers to extract features from the inputs. Then, the features are analyzed by the transformer block of the network. This approach ensures that the features provided to the attention mechanism capture information from all regions of an input. The convolutions also introduce inductive biases (*e.g.*, translational equivariance) into the network that normally require an extensive amount of data for a transformer to learn on its own. Convolution operations enable weight sharing, which decreases the size of the network and increases its computational efficiency. The CCT used in this work has approximately 405 thousand parameters, making it the smallest transformer we investigate by a significant margin. We train CCT from scratch with early stopping using a patience of 15 epochs. The AdamW optimizer [102] is used with an initial learning rate and weight decay of 10^{-4}. The batch size in our experiments

depends on the size of the mel spectrograms analyzed because the size of the attention matrix scales quadratically with the increase in input size [86]. For mel spectrograms sized 128x256, 128x512, and 128x1024, we use a batch size of 16, 16, and 2, respectively.

4.3.3 Patchout Fast Spectrogram Transformer (PaSST)

Patchout fast spectrogram transformer (PaSST) [95] is designed for analyzing mel spectrograms and considers frequency information explicitly. It has previously been used for audio classification [95] on Audio Set [103], which is a large-scale audio classification dataset with over 2 million sound clips. PaSST divides a 2-D mel spectrogram into smaller patches. Next, both frequency and time positional encodings are added to each patch so that the model knows the temporal and frequency ranges the patch represents. Finally, the patches are passed through the transformer. PaSST uses a technique called patchout to exclude certain patches during training. Patchout both shortens the input length (because fewer patches are analyzed) and regularizes the network (by forcing the network to succeed even when parts of the audio signals are missing). This reduces the network's reliance on certain frequency and temporal segments so that it can generalize to new audio signals better. The PaSST model used in our experiments has approximately 85.3 million parameters. We use an initial learning rate of 10^{-5} and weight decay of 10^{-4} with the AdamW optimizer [102] to train PaSST from scratch. Training occurs for 51 epochs with a batch size of 12. We use a patch stride of 10 both in time and frequency.

4.3.4 Self-Supervised Audio Spectrogram Transformer (SSAST)

Self-supervised audio spectrogram transformer (SSAST) [96] is another transformer that has been used for audio classification, and it is modeled after the audio spectrogram transformer [101]. Similar to PaSST, SSAST divides a 2-D mel spectrogram into patches and uses patch-based dropout to regularize the network. Because large amounts of labeled data are not always readily available for a specific task, SSAST proposes a self-supervised learning stage to use unlabeled data to learn general characteristics of a certain data distribution. After SSAST completes the self-supervised learning stage, it is fine tuned on labeled data

for a specific task. However, our experiments indicate that the self-supervision stage is not necessary for our task. We obtain better synthesized speech detection by training SSAST from scratch on our experimental dataset. We use an initial learning rate of 10^{-4} and weight decay $5 * 10^{-7}$ with the Adam optimizer [78] to train SSAST for 50 epochs with a batch size of 48, 48, and 12 for mel spectrograms sized 128x256, 128x512, and 128x1024, respectively. This network has approximately 87 million parameters.

4.3.5 Transformer Ensemble

We fuse the probabilities produced by each transformer to create the transformer ensemble. We explore two fusion techniques: averaging and maximizing. For these techniques, either the average probability or the maximum probability of all transformers are used as the final ensemble probability. Results indicate that the averaging technique detects synthesized speech better, so we report those results in this chapter. We also explore an ensemble using only the two best-performing transformers. Results indicate that the two-transformer ensemble is better than the three-transformer ensemble, so we report those results in this chapter. The transformers used in the transformer ensemble are CCT and PaSST, so we refer to the transformer ensemble as "CCT-PaSST".

4.4 Experimental Setup

4.4.1 Dataset

We again utilize the ASVspoof2019 dataset [79] in our experiments. We provide the training, validation, and testing breakdown in Table 4.1 for reference. Please refer to Chapters 2 and 3 for more information on the dataset.

4.4.2 Evaluation Metrics

For all experiments, we report Receiver Operating Characteristic Area Under the Curve (ROC AUC) and Precision Recall Area Under the Curve (PR AUC). Recall that for a binary classification problem (such as this synthesized speech detection task), a threshold

Table 4.1. ASVspoof2019 Dataset. Details about the dataset used in our experiments.

ASVspoof2019 Dataset					
Subset	Genuine Speech Signals	Synthesized Speech Signals	Total Speech Signals	Female Speakers	Male Speakers
Training	2,580	22,800	25,380	12	8
Validation	2,548	22,296	24,844	6	4
Testing	7,355	63,882	71,237	27	21
Total	12,483	108,978	121,461	45	33

must be selected to use as a cutoff to convert output probabilities to discrete categories (*i.e.*, *synthesized* or *authentic*). ROC AUC and PR AUC summarize detection performance using a full range of thresholds. Rather than reporting accuracy, precision, recall, and F1 metrics for a specific threshold, we wish to evaluate all methods based on their "robustness", or independence to a specific threshold. If ROC AUC and PR AUC are high, the detection method is able to detect synthesized speech for a large range of thresholds.

ROC AUC indicates how skillful (*i.e.*, successful) a detection method is. However, it does not account for class imbalances. If there are significantly more samples of one class in the evaluation set compared to the other class (as is the case with our dataset), it is possible for a detection method to have a high ROC AUC even if the detection method predicts the majority class all the time (but never predicts the minority class). For imbalanced datasets, it is important to consider other metrics, such as precision and recall, which measure how often the detection method correctly predicts the minority class. Thus, PR AUC is an important evaluation metric for our task because it will reflect the skill of all methods on our imbalanced dataset.

4.4.3 Experimental Results

Table 4.2 summarizes our experimental results. Results indicate that each transformer independently performs well on this task. Each individual transformer can achieve ROC AUC above 0.95 for at least one mel spectrogram size. CCT achieves the highest ROC

AUC of the individual transformers for all mel spectrogram sizes. All of the ROC AUC scores for CCT are above 0.96. PaSST has the second-highest ROC AUC scores, which are all above 0.92. However, the CCT-PaSST ensemble surpasses both CCT and PaSST ROC AUC scores. The CCT-PaSST ensemble consistently achieves ROC AUC higher than 0.98 for all mel spectrogram sizes.

Table 4.2. Results for Synthesized Speech Detection. This table shows results obtained with each method for each mel spectrogram size.

Synthesized Speech Detection Results			
Mel Spectrogram Size	Method	ROC AUC	PR AUC
128x256	CCT	0.9668	0.6446
	PaSST	0.9483	0.7332
	SSAST	0.9536	0.6774
	CCT-PaSST	**0.9810**	**0.8507**
128x512	CCT	0.9742	0.6876
	PaSST	0.9258	0.7012
	SSAST	0.7541	0.2213
	CCT-PaSST	**0.9816**	**0.8508**
128x1024	CCT	0.9718	0.7512
	PaSST	0.9590	0.7551
	SSAST	0.9064	0.4265
	CCT-PaSST	**0.9801**	**0.8585**

The PR AUC has more variability. Individual transformers achieve PR AUC ranging from 0.2213 to 0.7551. In this case, PaSST achieves the highest PR AUC of the individual transformers for all mel spectrogram sizes. Its PR AUC is greater than 0.70 for all mel spectrogram sizes. CCT has the second-highest PR AUC scores, which are all above 0.64 for all mel spectrogram sizes. Again, though, the CCT-PaSST ensemble achieves the best PR AUC scores. Its PR AUC is higher than 0.85 for all mel spectrogram sizes, which is higher than all PR AUC scores achieved with individual transformers by a significant margin.

Finally, let us consider how each method is impacted by mel spectrogram size. Each individual transformer seems sensitive to mel spectrogram size. CCT ROC AUC is consistent for all mel spectrogram sizes, but its PR AUC increases from 0.6446 to 0.7512 when the mel spectrogram length increases from 256 to 1,024. PaSST ROC AUC ranges from 0.9258 to 0.9590 as mel spectrogram sizes change, and its PR AUC ranges from 0.7012 to 0.7551 for different sizes. SSAST shows the most variability to mel spectrogram sizes, with its ROC AUC ranging from 0.7541 to 0.9536 and its PR AUC ranging from 0.2213 to 0.6774. However, the CCT-PaSST ensemble achieves consistent results for all mel spectrogram sizes. Its results do not even vary a percentage. The ROC AUC ranges from 0.9801 to 0.9816, and the PR AUC ranges from 0.8507 to 0.8585. Thus, our transformer ensemble consistently and successfully detects synthesized speech from all mel spectrogram sizes.

Figure 4.2 shows the ROC and PR curves for all methods trained and evaluated on mel spectrograms sized 128x256. From this figure, we see that all detection methods achieve high ROC curves that are fairly similar. The PR curves are more distinct, though. Clearly from Figure 4.2, we see that the CCT-PaSST ensemble outperforms all individual transformers. Thus, from both a visual and a numeric analysis, we conclude that our proposed transformer ensemble achieves the best synthesized speech detection.

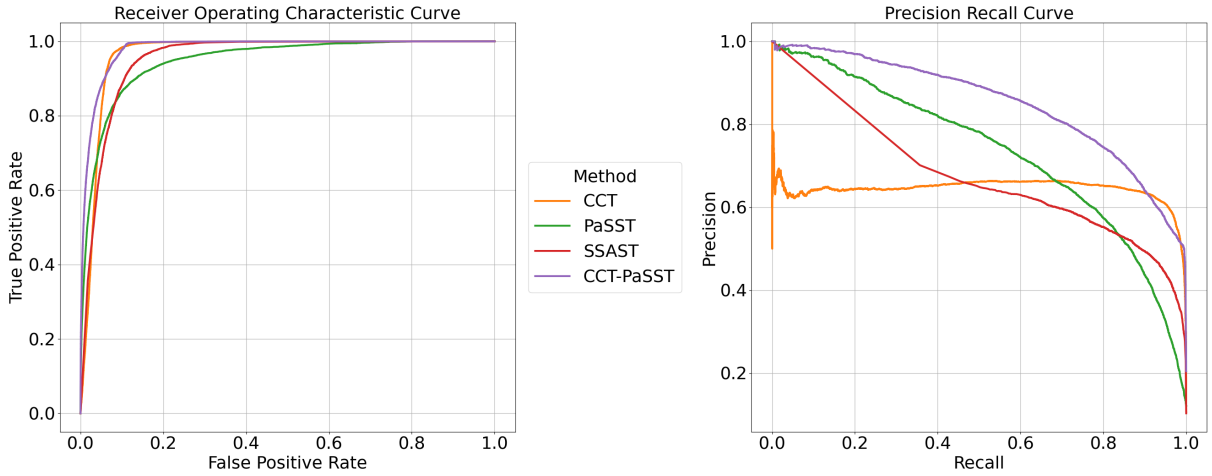


Figure 4.2. ROC and PR Curves for Synthesized Speech Detection. Results obtained with each method analyzing mel spectrograms sized 128x256. The CCT-PaSST transformer ensemble achieves the best synthesized speech detection results, especially in terms of PR AUC.

5. SYNTHESIZED SPEECH ATTRIBUTION WITH A TRANSFORMER

5.1 Overview

Many free, easy-to-use tools offer text-to-speech or voice conversion utilities to generate synthetic speech [27], [28], [34]–[37]. Synthesized speech can provide many benefits to society, such as accessibility services to support people who are visually impaired. However, synthesized speech can also be used for malicious purposes. In 2021, an impersonator using synthesized speech on a conference call tried to convince Goldman Sachs to make a $40 million investment [43]. In 2022, a deepfake video showed Ukrainian President Volodymyr Zelensky surrendering to Russia [41]. A recent AI-produced podcast contains a realistic-sounding conversation between Steve Jobs and Joe Rogan, but all of the speech is synthesized [104]. These examples highlight the importance of audio forensics. Large amounts of synthesized speech may be generated for large-scale scams and disinformation campaigns. The same speech synthesizers are often utilized throughout the scams. Thus, identifying the speech synthesizer used to create speech signals in these campaigns can reveal more information about how the scams are spreading and (possibly) who created them.

In this chapter, we investigate speech synthesizer attribution, which is the task of identifying which speech synthesizer was used to generate a synthetic speech signal. To attribute a speech signal to a speech synthesizer, we convert speech signals into spectrograms[105]. A spectrogram is a 2-D temporal-spectral representation of a speech signal. We treat the spectrograms as images and analyze them with our proposed deep learning method, called compact attribution transformer (CAT). We consider eleven different speech synthesizers in the scope of this work. Eight of the synthesizers are used to train CAT. The remaining three synthesizers are only used during testing to represent a set of unknown speech synthesizers that have not been seen during training. We evaluate our method on both a closed set of known speech synthesizers as well as an open set, which contains a combination of known and unknown synthesizers. We demonstrate that our proposed approach successfully attributes synthesized speech signals to their synthesizers and can differentiate between known and unknown synthesizers. We utilize a t-distributed stochastic neighbor embedding (tSNE) [106]

to separate synthesized speech signals created by different synthesizers in CAT's latent space. We show that tSNE successfully discriminates different synthesizers, including different unknown synthesizers. Thus, our proposed approach generalizes to new synthesizers. Finally, we improve attribution performance by using poly-1 loss formulations.

5.2 Related Work

Most audio forensics methods focus on detecting manipulated or synthesized speech [88], [98], [107], [108]. However, identifying the source of the audio is also important. Different recording devices, editing software, and file compression methods leave clues about the nature of an audio signal. It is possible to associate – or attribute – authentic audio signals to specific components used in their creation. For example, Buchholz *et al.* identify microphones used to record audio signals by analyzing the signals' Fourier coefficients [109] with naïve bayes, a SVM, logistic regression (LogReg), decision trees, and a KNN. Luo *et al.* focus specifically on smartphone attribution for audio recordings [110]. They propose a feature set called a bank energy difference descriptor (BED) for analysis with a SVM. Just as it is possible to attribute an authentic speech signal to a specific audio-capturing device, it is possible to identify speech synthesizers used to generate speech signals because each speech synthesizer leaves its own fingerprints in the audio signals it creates. Borrelli *et al.* explore speech synthesizer attribution using a random forest, a linear SVM, and a non-linear SVM in both closed set and open set scenarios [111].

In this work, we propose a deep learning method know as compact attribution transformer (CAT) for speech synthesizer attribution. CAT is a convolutional transformer, meaning that it utilizes a series of convolutional layers before a transformer to train more efficiently and incorporate inductive biases into the network. A convolutional transformer has achieved success in synthetic audio detection [99], so now we explore its use for synthetic speech attribution.

Besides using deep learning, our approach differs from prior audio attribution work in that it analyzes spectrograms of audio signals. We use spectrograms to leverage frequency information of speech signals. Because spectrograms provide a structured version of a speech

signal and group similar frequencies together, CAT can analyze different frequency ranges more explicitly to identify artifacts of various speech synthesizers. Another novel aspect of our work is the use of t-distributed stochastic neighbor embedding (tSNE) [106] to discriminate between different unknown synthesizers. TSNE is an unsupervised learning method used for dimensionality reduction and visualizing high-dimensional data. We use tSNE on the latent space of CAT to separate all known and unknown synthesizers. Finally, we explore different loss functions to improve performance of the method. We formulate the loss function used in CAT as poly-1 losses to tailor the loss function to this specific task. We show that our proposed approach achieves success on this attribution task, even in an open set scenario.

5.3 Proposed Approach

Figure 5.1 shows an overview of our approach. First, we convert all speech waveforms into normalized spectrograms [105]. Next, we use a series of convolutional layers (*i.e.*, a convolutional block) to extract feature maps from the spectrograms. Then, a transformer analyzes the feature maps and produces a set of probabilities $P = \{p_i \mid 0 \leq i \leq N-1\}$, where N corresponds to the number of known synthesizers, and probability p_i indicates the likelihood that a speech signal under analysis was generated by speech synthesizer i. We use the probabilities in P to determine which of the known speech synthesizers created the speech signal. We also use a thresholding procedure on the probabilities to determine whether a speech signal was generated by a known or an unknown speech synthesizer.

5.3.1 Spectrogram Creation

We convert speech waveforms into spectrograms by using a 32 ms Hanning window on the speech signals with a shift of 8 ms. In other words, we use the Fast Fourier Transform (FFT) on blocks of the signals consisting of 512 sampled points with 128 points of overlap between consecutive blocks [105]. Note that all waveforms in our experimental dataset have a 16 kHz sampling rate. Next, the FFT coefficients are converted to decibels and organized into 2-D arrays, where height corresponds to frequency bands and width corresponds to time (*i.e.*,

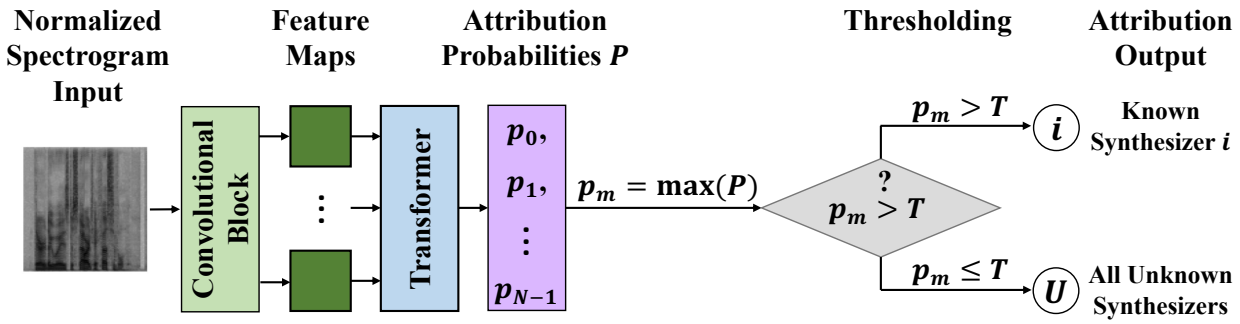


Figure 5.1. Block Diagram of our Compact Attribution Transformer (CAT). CAT is a convolutional transformer that analyzes spectrograms sized 128x128. After producing a set of attribution probabilities P with the transformer block, CAT uses a thresholding procedure to determine if the speech signal was created by a known or an unknown speech synthesizer.

length of the audio signal in terms of 32 ms time blocks). We crop or pad each spectrogram to have dimensions 128x128 pixels to ensure that all inputs to the neural network are the same size. Finally, we normalize the spectrogram pixels to the range of values [0,1]. Normalized values enable machine learning models to learn more quickly because they are forced to focus on relative rather than absolute differences in input values. The normalized spectrograms of size 128x128 pixels are used as inputs to CAT.

5.3.2 Compact Attribution Transformer (CAT)

We propose a method called the compact attribution transformer (CAT) for speech synthesizer attribution. It is a convolutional transformer that first extracts feature maps from an input spectrogram using convolutions. Then, it uses an attention mechanism [85] to analyze the feature maps in order to attribute the audio signal under analysis to a speech synthesizer. One significant benefit of our approach compared to other transformers is its size. We design CAT to have 405 thousand parameters. There are a number of other transformers that have demonstrated success in image-based analyses, but they are significantly larger in size. For example, the first transformer used on images – known as vision transformer (ViT) [83] – has three variants of different sizes: ViT-Base has 86 million parameters; ViT-Large has 307 million parameters; and ViT-Huge has 632 million parameters. ViT was adapted for spectrogram analysis to produce a model called audio spectrogram transformer

(AST) [101]. There are other variations of AST, such as self-supervised audio spectrogram transformer (SSAST) [96], that use self-supervised learning during training for audio classification. Because AST and its variants (*e.g.*, SSAST) are based on ViT, they too have over 87 million parameters. Patchout fast spectrogram transformer (PaSST) [95] is another transformer used to analyze spectrograms for audio classification. It explicitly uses frequency information in its attention mechanism and has over 85 million parameters. Because CAT uses significantly fewer parameters in comparison to these transformers, it trains faster and is easier to work with in scenarios with limited compute power.

Another benefit of CAT is that it does not require an extremely large dataset for training [86]. This is important for our work because we investigate speech synthesizer attribution with a relatively small dataset containing only 17,000 samples (see Section 5.4 for details of the dataset). Transformers lack some of the inductive biases that convolutions introduce, such as translational equivariance and locality [83], [86], and must learn these properties on their own. However, this requires larger datasets that exhibit the properties. Larger datasets are not always available for particular tasks (*e.g.*, medical tasks, our speech synthesizer attribution task), making it difficult to train transformers properly. CAT explicitly introduces these biases into the network by using a convolutional block before the transformer. The convolutional block also increases the parameter efficiency of CAT compared to non-convolutional transformers by using shared weights. This enables CAT to train more efficiently with less data.

CAT is based on a network known as compact convolutional transformer [86]. Our CAT network uses two convolutional layers in the convolutional block. Each convolutional layer uses a 3x3 kernel, ReLU activation functions, max pooling, and positional embedding. The transformer encoder contains two transformer layers that use layer normalization, two attention heads, GeLU activation functions, stochastic depth, and sequence pooling [86]. The final dense layer of the network uses a softmax function to create the set of output probabilities P. We train CAT for 100 epochs with a patience of 10 epochs and use the AdamW optimizer [102] with an initial learning rate and weight decay of 10^{-4}. The batch size in our experiments is 128.

5.3.3 Loss Functions

We investigate the use of three different loss functions in this work. First, we use a standard cross entropy (CE) loss [92]. Then, we use a poly-1 cross entropy loss (poly-1-CE) and a poly-1 focal loss (poly-1-FL) [112]. Poly-1 losses approximate loss functions via Taylor expansion. This allows loss functions to be designed as linear combinations of polynomial functions with easily adjusted polynomial bases. The polynomial bases can be customized for a specific dataset or specific task, enabling better performance with the same dataset and model architecture.

Let us begin with the definition of cross entropy (CE) loss:

$$L_{CE} = -log(p_i). \tag{5.1}$$

The poly-1 cross entropy loss is then:

$$L_{poly-1-CE} = L_{CE} + \epsilon(1 - p_i). \tag{5.2}$$

Similarly, we can formulate the poly-1 focal loss. Let us begin with the definition of focal loss (FL):

$$L_{FL} = -(1 - p_i)^{\gamma} log(p_i) \tag{5.3}$$

where γ serves as a modulating factor. It reduces the loss contribution from "easy" data samples by extending the range of probabilities that contribute to a low loss value. Note that when $\gamma = 0$, focal loss is equivalent to cross entropy loss (*i.e.*, $L_{FL} = L_{CE}$). Then, the poly-1 focal loss is:

$$L_{poly-1-FL} = L_{FL} + \epsilon(1 - p_i)^{\gamma+1}. \tag{5.4}$$

By constructing the loss function with this poly-1 formulation, we can adapt the loss function for our specific dataset by changing ϵ. We perform a gridsearch to find the best value of ϵ for our task. Experiments indicate that $\epsilon = 3.3$ yields the best results with poly-1 cross entropy loss, and $\epsilon = 3$ yields the best results with poly-1 focal loss. Note that we use $\gamma = 2$ with poly-1 focal loss. More details of these experiments are included in Section 5.4.

5.3.4 Identifying Unknown Synthesizers

In addition to attributing synthesized speech signals to known synthesizers, we also determine if a speech signal is created by an unknown synthesizer. To do so, we implement a thresholding procedure on the probabilities in P produced by CAT. We define a threshold T that determines if CAT is confident in its attribution or not. Let p_m represent the maximum probability in P (*i.e.*, $p_m = \max(P)$). If $p_m > T$, then CAT is very confident that it can attribute the speech signal under analysis to the known synthesizer i. If none of the probabilities in P are greater than the threshold T, CAT cannot detect any fingerprints from any of the known synthesizers. Instead, the signal must be created by an unknown synthesizer. In this case, it is assigned to the category containing all unknown synthesizers U. In practice, any threshold T can be selected depending on the application and dataset.

This thresholding procedure enables attribution to $N + 1$ classes (*i.e.*, to one of the known synthesizers or to the unknown category U). However, it does not allow for discrimination between different unknown synthesizers within U. To attribute a synthesized signal to a specific unknown synthesizer, we utilize a t-distributed stochastic neighbor embedding (tSNE) [106] on the latent space of CAT. First, we embed all speech signals in the open set into the latent space of CAT by running them through the model. Next, we use tSNE on the output of CAT's transformer block, which is the layer directly before the final dense layer that maps the latent space of CAT to a lower-dimensional output vector of length N (*i.e.*, the number of known synthesizers). TSNE projects the high-dimensional latent space of CAT onto two dimensions, allowing visualization of CAT's internal representation of all of the speech signals in the open set. Then, we examine the tSNE visualization to understand how CAT separates different synthesizers in its latent space. Our results indicate that distinct clusters form in the tSNE visualization and that each cluster corresponds to a different synthesizer. Even the different unknown synthesizers have distinct clusters in the latent space. Thus, analyzing the tSNE visualization provides more details about the nature of synthesized speech signals. We present our findings in Section 5.4. In our experiments, we use a perplexity of 50 and 1,500 iterations to fit tSNE to CAT's latent space.

5.4 Experimental Setup

5.4.1 Dataset

To validate our approach, we use a dataset known as the SemaFor Audio Model Attribution Dataset. It was presented in the Semantic Forensics (SemaFor) program [47] organized by DARPA. The dataset is based on a dataset from the 2022 IEEE Signal Processing Cup (SP Cup) [113], which is a competition related to synthetic speech attribution. The dataset from the SP Cup was augmented with another speech synthesizer (known as Riva [114]) and re-arranged to create the final version of the SemaFor Audio Model Attribution Dataset.

There are a total of 17,000 synthesized speech signals in the dataset from 11 different speech synthesizers. Speech signals from eight synthesizers are considered to be "known" (*i.e.*, part of the closed set). Speech signals from the remaining three synthesizers are considered to be "unknown" (*i.e.*, part of the open set). The speech synthesizers in the closed set are FastPitch[115], FastSpeech2 [116], Glow-TTS [117], gTTS [118], Tacotron [119], Tacotron 2 [120], TalkNet [121], and Riva [114]. The speech synthesizers in the open set are Mixer-TTS [122], SpeedySpeech [123], and VITS [34]. The speech synthesizers use various deep learning methods, including transformers [85], generative flow models [124], [125], CNNs [126], LSTMs [81], [127], MLPs [84], GANs [82], [128], [129], and variational autoencoders (VAEs) [130] to generate new speech signals. All signals have a sampling rate of 16 kHz.

Table 5.1 summarizes the details of the dataset. We utilize the official dataset split according to the SemaFor program for training and testing our approach. Notice that there are an unequal number of samples associated with each speech synthesizer. For example, Tacotron 2 and FastSpeech2 only have 500 training samples, but the other speech synthesizers in the closed set have 1,000 training samples. Similarly, these two synthesizers have only 300 testing samples, while some other synthesizers have 1,500 testing samples. This imbalance presents a challenge in training an attribution model because there are fewer data samples that can be used to learn to recognize certain speech synthesizers. One other challenge with this dataset is that not all speech synthesizers use the same speaker (*i.e.*, simulate the same voice). Most of the speech synthesizers replicate a speaker from the LJSpeech

dataset [131], but some synthesizers use speakers from the LibriSpeech dataset [132], the VCTK dataset [133], and an unknown dataset used by Google. Because there are different speakers in the dataset, an audio attribution model must learn the characteristics of a speech synthesizer regardless of the speaker.

Table 5.1. SemaFor Audio Model Attribution Dataset. Dataset used in our experiments.

SemaFor Audio Model Attribution Dataset					
Speech Synthesizer	Training Samples	Testing Samples	Total Samples	Speakers	Type
FastPitch	1,000	1,500	2,500	1 from LJSpeech	Known
FastSpeech2	500	300	800	1 from LJSpeech	Known
Glow-TTS	1,000	1,500	2,500	1 from LJSpeech	Known
gTTS	1,000	1,500	2,500	4 unknown	Known
Tacotron	1,000	1,500	2,500	10 from LibriSpeech	Known
Tacotron 2	500	300	800	1 from LJSpeech	Known
TalkNet	1,000	1,500	2,500	1 from LJSpeech	Known
Riva	1,000	1,000	2,000	1 from LJSpeech	Known
Mixer-TTS	0	300	300	1 from LJSpeech	Uknown
SpeedySpeech	0	300	300	1 from LJSpeech	Uknown
VITS	0	300	300	10 from VCTK	Uknown
Total	7,000	10,000	17,000	25	

5.4.2 Evaluation Process

We report accuracy, weighted precision, weighted recall, and weighted F1 [93] of all methods evaluated. Weighted metrics are computed with a weighted average of each metric obtained on each of the classes in the dataset, where weights reflect the dataset class imbalance. Recall that all evaluated methods produce a set P of N probabilities, where each probability represents the likelihood that a known synthesizer was used to generate a speech signal under analysis. In a closed set scenario, the final attribution output is the synthesizer that corresponds to the maximum probability $p_m \in P$. All metrics are computed based on only the N known synthesizers. In an open set scenario, the thresholding procedure pro-

duces the final attribution output, which could either be one of the known synthesizers or an unknown class U. In this case, all metrics are computed based on the $N + 1$ classes.

5.4.3 Baselines and Comparison Methods

To validate our approach, we compare it against theoretical baselines, classical machine learning methods, and deep learning methods. We refer to our theoretical baselines as Baseline-Minority and Baseline-Majority. These baselines represent theoretical classifiers that only predict one synthesizer. In other words, every time a new speech signal is presented to the theoretical classifiers, they always predict the same attribution output. In the case of Baseline-Minority, the output is always the minority class (*i.e.*, the dataset class that corresponds to the fewest samples in the training set). In the case of Baseline-Majority, the output is always the majority class (*i.e.*, the dataset class that corresponds to the most samples in the training set). In the case where multiple classes have the same number of data samples, one of the tied classes is chosen to represent the minority/majority class. Because these baselines are computed from the experimental dataset, they establish lower bounds of the evaluation metrics to indicate whether a classifier actually performs well.

In addition to these baselines, we investigate several classical machine learning methods: quadratic discriminant analysis (QDA); gaussian process (GP); AdaBoost; k-nearest neighbors (KNN); naïve bayes; decision tree; random forest; non-linear and linear support vector machines (SVMs); and logistic regression (LogReg). To construct the inputs to these methods, the normalized spectrograms of size 128x128 pixels are row concatenated to produced flattened, 1-D arrays of length 16,384. Each of these methods has various hyperparameters that can affect their performance, so we execute a gridsearch to determine the best set of hyperparameters for each method. Table 5.5 specifies the best hyperparameters for all methods included in our analysis.

Finally, we explore two deep learning methods in addition to our proposed CAT solution: a MLP and a CNN. The MLP operates on the 1-D version of the spectrograms, while the CNN operates on the original 2-D format (*i.e.*, the same format analyzed by CAT). The MLP consists of two hidden layers of 1,500 nodes with logistic activation functions and one

output layer that consists of 8 nodes (equal to the number of known synthesizers in our experimental dataset). We train the MLP for 200 epochs with a patience of 10 epochs using the Adam optimizer [78] with an initial learning rate of 10^{-4}. The batch size is 200 in our experiments. The CNN consists of two convolutional layers followed by two dense layers. The convolutional layers use a 3x3 kernel and ReLU activation functions. The first dense layer contains 128 nodes, and the second dense layer (*i.e.*, the output layer) contains 8 nodes. The CNN uses max pooling and dropout to regularize the network. We train the CNN for 100 epochs with a patience of 10 epochs using a batch size of 128. The CNN uses the Adam optimizer [78] with an initial learning rate of 10^{-3}. Both the MLP and the CNN use a softmax activation function on the outputs of the final dense layer to create a set of output probabilities P. They also both use cross entropy as a loss function.

5.4.4 Experimental Results

Table 5.2. CAT Results on Closed Set with Different Losses. Results reported in this table are the best results achieved with each poly-1 loss function after a search for the poly-1 hyperparameter term ϵ.

CAT Results with Different Loss Functions				
Loss Function	Accuracy	Weighted Precision	Weighted Recall	Weighted F1
L_{CE}	90.12%	89.05%	90.12%	89.45%
$L_{poly-1-CE}$	**92.53%**	**90.37%**	**92.53%**	**91.27%**
$L_{poly-1-FL}$	92.02%	89.65%	92.02%	90.67%

Table 5.2 shows results with CAT using different loss functions. The baseline loss function is cross entropy L_{CE}, which achieves approximately 90% accuracy, weighted precision, weighted recall, and weighted F1 on the closed set. Both poly-1 loss formulations improve attribution results compared to this baseline. Poly-1 focal loss ($L_{poly-1-FL}$) improves all results, especially in terms of accuracy and weighted recall. Poly-1 cross entropy loss ($L_{poly-1-CE}$) improves results even further. All metrics increase by roughly 2% with $L_{poly-1-CE}$, result-

ing in 92.53% accuracy and 91.27% F1. Because poly-1 cross entropy achieves the greatest results overall, CAT experiments reported in the rest of this chapter use $L_{poly-1-CE}$.

Table 5.3 shows results of all methods on the closed set. All methods outperform the theoretical baselines and demonstrate that they achieve greater attribution success than minority class and majority class classifiers. However, some methods cannot even achieve 50% accuracy on this attribution task. For example, all metrics obtained with QDA, GP, and AdaBoost are less than 35%. KNN barely passes the 50% mark. Thus, not all methods are suited for synthesizer attribution. Decision trees achieve better success on this task, obtaining approximately 70% accuracy. Using an ensemble of decision trees (*i.e.*, a random forest) improves results even further to ∼80% accuracy. SVMs achieve better attribution performance, with both linear and non-linear SVMs achieving roughly 81% accuracy on this task. LogReg achieves the best performance of the classical machine learning methods (∼90%). However, two deep learning methods achieve better performance. Both CNN and CAT achieve 90% or higher for all metrics considered. Our CAT with poly-1 cross entropy loss outperforms all methods for all metrics, achieving ∼93% accuracy and ∼91% F1. Figure 5.3 shows more details about the performance of CAT on the closed set. This confusion matrix indicates that CAT can attribute speech signals to most of the known synthesizers (7 out of 8) very well. CAT struggles with just one synthesizer: Tacotron 2. Tacotron 2 is one of the synthesizers with the fewest training and testing samples. It is more difficult for an attribution method to learn to identify a synthesizer from fewer samples, which is one factor contributing to the challenge in identifying Tacotron 2 speech signals.

Table 5.4 shows results on the open set using the thresholding method. In general, attribution results drop for all approaches in the open set scenario. One method's performance (GP) even drops below Baseline-Majority. Other methods, such as QDA and AdaBoost, barely beat Baseline-Majority based on some metrics. Most methods' metrics drop to under 80%. Only three methods achieve metrics greater than 80%: LogReg, CNN, and CAT. This drop in performance indicates the difficulty of synthesizer attribution in an open set scenario. However, CAT is still able to attribute synthesized speech with over 84% accuracy and 83% F1, achieving the highest attribution results overall. Thus, results indicate that CAT with

Table 5.3. Results of All Methods on Closed Set. Results reported in this table are the highest metrics obtained with each method from a hyperparameter search.

Results of All Methods on the Closed Set				
Method	Accuracy	Weighted Precision	Weighted Recall	Weighted F1
Baseline-Minority	3.30%	0.11%	3.30%	0.21%
Baseline-Majority	16.48%	2.72%	16.48%	4.67%
QDA	19.34%	16.82%	19.34%	11.75%
GP	21.62%	68.80%	21.62%	12.82%
AdaBoost	33.58%	32.71%	33.58%	23.69%
KNN	52.22%	56.63%	52.22%	49.83%
Naïve Bayes	68.14%	70.95%	68.14%	69.08%
Decision Tree	69.02%	71.08%	69.02%	69.93%
MLP	78.91%	77.06%	78.91%	77.68%
Random Forest	81.04%	79.90%	81.04%	79.10%
Non-Linear SVM	81.13%	81.35%	81.13%	81.05%
Linear SVM	81.57%	80.99%	81.57%	81.22%
LogReg	90.68%	88.29%	90.68%	89.43%
CNN	91.99%	90.21%	91.99%	90.88%
CAT	**92.53%**	**90.37%**	**92.53%**	**91.27%**

poly-1 cross entropy loss is the best method for synthesized speech attribution in both closed set and open set scenarios.

Figure 5.2 shows CAT's latent space representation of all speech signals in the open set. We explore the tSNE plot to better understand the attribution performance of CAT. It shows distinct clusters for most of the synthesizers in our experimental dataset. For example, gTTS, Glow-TTS, Tacotron, Riva, FastSpeech2, SpeechSpeech, VITS, and Mixer-TTS all have their own clusters within the latent space. This separability indicates that CAT can successfully identify unique fingerprints of each of these synthesizers and use them for attribution. Recall that Mixer-TTS, SpeedySpeech, and VITS are all unknown synthesizers.

Table 5.4. Results of All Methods on Open Set. Results reported in this table are the highest metrics obtained with each method from a hyperparameter search.

Results of All Methods on the Open Set				
Method	Accuracy	Weighted Precision	Weighted Recall	Weighted F1
Baseline-Minority	3.00%	0.09%	3.00%	0.17%
Baseline-Majority	15.00%	2.25%	15.00%	3.91%
QDA	17.60%	14.45%	17.60%	9.86%
GP	9.00%	0.81%	9.00%	1.49%
AdaBoost	17.51%	10.84%	17.51%	12.80%
KNN	47.52%	46.89%	47.52%	42.77%
Naïve Bayes	62.01%	59.58%	62.01%	59.95%
Decision Tree	62.66%	60.01%	62.66%	60.77%
MLP	55.97%	57.03%	55.97%	53.36%
Random Forest	47.74%	80.92%	47.74%	46.29%
Non-Linear SVM	65.41%	73.24%	65.41%	66.41%
Linear SVM	68.47%	71.78%	68.47%	68.80%
LogReg	83.85%	81.44%	83.85%	81.62%
CNN	83.56%	77.26%	83.56%	79.67%
CAT	**84.10%**	**82.37%**	**84.10%**	**83.00%**

Thus, CAT can actually discriminate between different unknown synthesizers, even though they have never been presented to the network before. This ability can provide further information to forensic analysts and help them in real-world scenarios as new synthesizers are invented. TSNE also provides further details about CAT's vulnerabilities. Although most synthesizers are separated into distinct clusters in the tSNE plot, three synthesizers have less separability: TalkNet, Tacotron 2, and FastPitch. The tSNE plot indicates that TalkNet and FastPitch have more separability from each other, but Tacotron 2 has extensive overlap with TalkNet and FastPitch. Thus, tSNE confirms results summarized by the confusion matrix in Figure 5.3. Figure 5.3 indicates that CAT is not able to attribute Tacotron 2 speech signals,

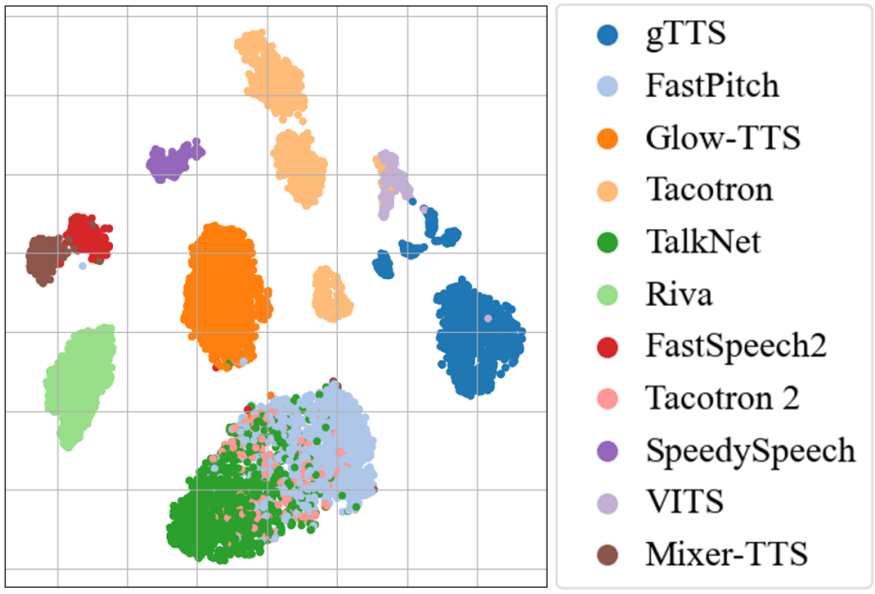


Figure 5.2. TSNE Plot of CAT's Latent Space on Open Set. Mixer-TTS, SpeedySpeech, and VITS are unknown synthesizers.

and Figure 5.2 provides evidence that CAT cannot separate those signals correctly. To try to mediate this issue, we explored weighted loss functions that account for the dataset's class imbalance. We also investigated methods that could discriminate between Tacotron 2 speech signals and all other synthesizers in a one-vs-all fashion. However, these initial investigations were unsuccessful at attributing Tacotron 2 signals. In future work, we seek to overcome this challenge.

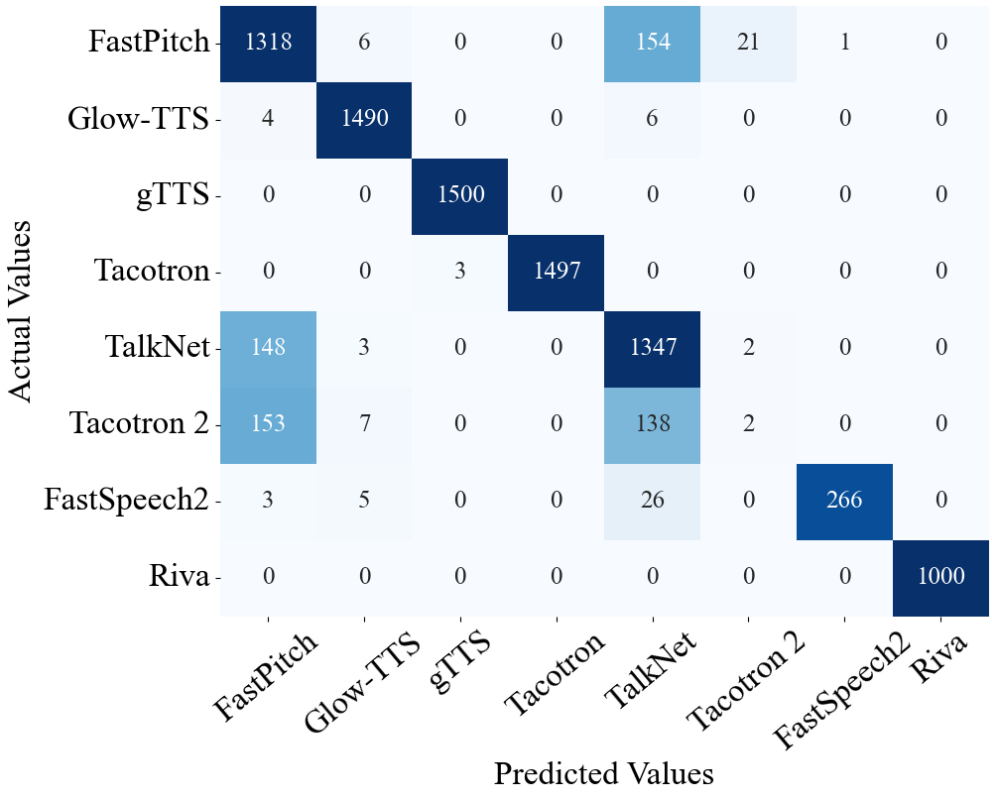


Figure 5.3. Confusion Matrix of CAT Results with poly-1-CE Loss. CAT can attribute speech signals to most synthesizers very well, but it struggles with attribution to Tacotron 2.

Table 5.5. Hyperparameters. Values used to achieve best results.

Method	Hyperparameter	Value
QDA	reg_param	0.0
GP	optimizer	fmin_l_bfgs_b
AdaBoost	n_estimators	50
KNN	n_neighbors	2
	weights	distance
	algorithm	auto
	leaf_size	10
	p	1
Naïve Bayes	var_smoothing	1e-7
Decision Tree	criterion	gini
	splitter	best
	min_samples_split	10
	class_weight	balanced
MLP	hidden_layer_sizes	(1500, 1500)
	activation	logistic
	batch_size	200
	epochs	200
	patience	10
	early_stopping	True
	optimizer	adam
	learning_rate_init	0.0001
	loss_function	CE
Random Forest	n_estimators	500
	min_samples_split	5
	class_weight	balanced
Non-Linear SVM	C	1
	kernel	poly
	degree	2
	gamma	scale
	shrinking	True
	class_weight	None
	probability	True
	max_iter	100
Linear SVM	C	0.1
	kernel	linear
	shrinking	True
	class_weight	None
	probability	True
	max_iter	100
LogReg	penalty	L1
	C	10
	class_weight	balanced
	solver	liblinear
	max_iter	500
	l1_ratio	0.75
CNN	batch_size	128
	epochs	100
	patience	10
	optimizer	adam
	learning_rate_init	0.001
	loss_function	CE
CAT	batch_size	128
	epochs	100
	patience	10
	optimizer	adamW
	learning_rate_init	0.001
	weight_decay_init	0.001
	loss_function	poly-1-CE
	ϵ	3.3

6. MACHINE LEARNING FOR HYPERSONIC VEHICLE CLASSIFICATION

6.1 Overview

Hypersonic vehicles fly at tremendous speeds. They travel faster than five times the speed of sound in high supersonic ranges between Mach 5 and Mach 10 (5,000 to 25,000 km/h) [134]. At peak velocities, hypersonic vehicles travel approximately five miles a second [135]. As a result of their speed, hypersonic vehicles could fly anywhere in the world in less one hour [136], [137].

One type of hypersonic vehicle is a hypersonic glide vehicle (HGV), which possesses maneuvering capabilities. Typically, rockets launch HGVs into the upper atmosphere. HGVs then travel in and above the atmosphere at altitudes ranging from 40 km to over 100 km. They glide to their final destinations by skipping off the top of the atmosphere [135], [138]. This bouncing action can be seen in Figures 6.1, 6.2, 6.3, and 6.10b, after a vehicle pulls up from its initial dive. These figures show normalized values since machine learning methods perform best on normalized data [139]–[142]. HGVs exploit aerodynamic forces to steer to their destinations, bypassing defense tracking systems and interceptors en route. Such maneuvering presents challenges in predicting their destinations. HGVs possess the capability to change their trajectories, and thus their final destinations, during flight [135].

Hypersonic glide vehicles distinguish themselves from conventional ballistic and cruise vehicles by coupling speed with maneuverability. Although ballistic vehicles can travel at hypersonic speeds, they do not maneuver. They follow relatively predictable hyperbolic flight paths to their final destinations. Cruise vehicles sacrifice speed for maneuverability. They travel slower than Mach 1 (*i.e.*, the speed of sound) at speeds less than approximately 1,000 km/h [135]. Hypersonic glide vehicles excel at both key characteristics, achieving ultra-high velocities with high mobility. Due to their advanced capabilities, HGVs may replicate behaviors exhibited by legacy vehicles initially in order to disguise their true nature and inhibit a successful defense response.

In this chapter, we investigate various machine learning methods to identify hypersonic vehicles and reentry vehicles based on their aerodynamic features. We train and evaluate our

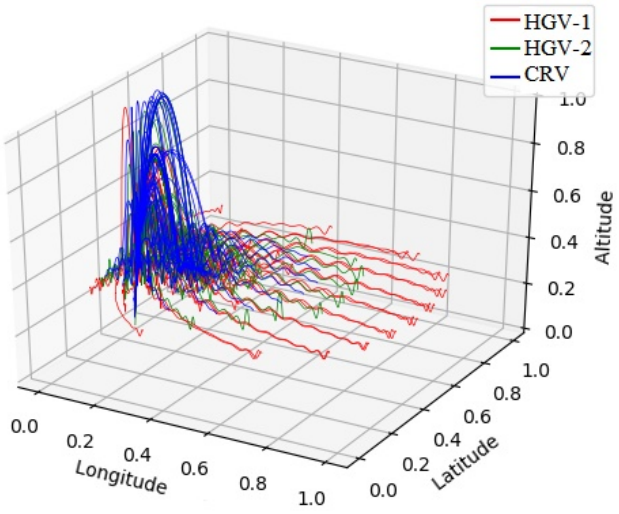


Figure 6.1. Vehicle Dataset Trajectories. This plot shows all trajectories in the dataset used for analysis based on their 3-D position coordinates.

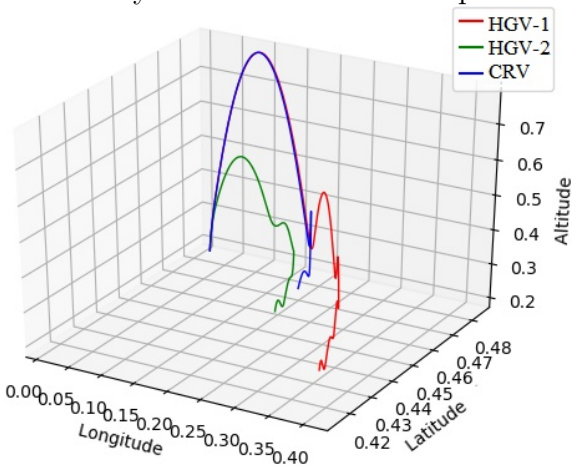


Figure 6.2. Example Trajectories from Each Vehicle Type. This plot shows a trajectory from each vehicle type.

methods on a multi-class dataset consisting of estimated states with notional radar quality of service portraying vehicle trajectories. We demonstrate that our approaches achieve good accuracy on this classification task and that accuracy increases with time after lift-off (TALO) of the vehicle.

6.2 Related Work

Various research techniques characterize vehicles. Some of these methods strive to differentiate ballistic vehicles from decoys, debris, and other airborne vehicles [143]–[145]. Other techniques focus on recognizing different subcategories of vehicles (*e.g.*, ballistic vehicles, reentry vehicles, quasi-ballistic vehicles, and cruise vehicles) [146]–[151]. Recently, a method emerged which classifies hypersonic glide vehicles in addition to ballistic vehicles [152].

In order to identify ballistic vehicles from other objects, some approaches utilize a machine learning method based on a KNN classifier [143]–[145]. Persico *et al.* extract feature vectors consisting of pseudo-Zernike moments from high-resolution range profile (HRRP) frames [143], [144]. They use a KNN classifier with the feature vectors to achieve vehicle discrimination. In another approach, Perisco *et al.* analyze features based on Krawtchouk moments with KNN used for classification [145]. Jithesh *et al.* also utilize HRRPs for vehicle classification [151]. Although they do not consider ballistic vehicles specifically, they employ a LSTM architecture to recognize different airborne vehicles.

For classifying different types of vehicles, many approaches use Bayesian models and machine learning methods. Farina *et al.* use a multiple model maximum likelihood estimator (MM-MLE) to classify vehicles based on kinematic features [150]. Park *et al.* recognize ballistic vehicles during their ascent phases with a thresholding method [149]. They obtain the thresholds through Monte-Carlo simulation. Singh *et al.* implement a Hidden Markov Model (HMM) to classify ballistic vehicles based on their kinematic attributes, including specific energy, acceleration, altitude, and velocity [148]. They also introduce a method that unites a HMM with a Real-Time Neural Network (RTNN) [146], [147].

Gaiduchenko *et al.* [152] propose a hypersonic convolutional neural network (HCNN) to classify both hypersonic and ballistic vehicles. They describe a method that analyzes flight path data consisting of 3-D coordinates which describe the position of the vehicles under analysis during flight. The HCNN only requires position data in order to classify whether a trajectory is that of a hypersonic glide vehicle, a hypersonic cruise missile, or a conventional ballistic missile.

Our proposed approach also utilizes machine learning methods to classify vehicles. We examine different aerodynamic features than previous methods. More specifically, our methods analyze angle of attack, angle of attack rate, and dynamic pressure. Our experiments focus on hypersonic vehicles specifically and include analysis of multiple HGVs. We demonstrate that our method successfully classifies HGV and RV trajectories based on key vehicle maneuvers and that accuracy increases with TALO.

6.3 Initial Work

Prior to developing and finalizing our methodology presented in Section 6.4, extensive experimentation, investigation, analysis, and discussion was conducted on other methods. This initial work explored MLP, LSTM, and CNN architectures. The experiments also explored the use of different aerodynamic features analyzed by the ML networks and evaluated vehicle data in a 2-class, 3-class, and 4-class fashion. This section discusses the initial work that helped refine the ultimate method utilized.

6.3.1 Initial Dataset

To construct the dataset used in these experiments, we simulate altitude, longitude, and latitude at a 1 Hz sampling rate for entire trajectories. The trajectories and associated features are normalized in order to enable better learning of the machine learning models [139]–[142]. Our dataset contains trajectories resulting from various launch configurations. Each launch configuration is defined by its start location and rocket stages. For our experiments, start positions include the ground and the air. In the case of an air launch, vehicles are released from another aircraft that already achieved a desire altitude. The number of rocket stages refers to the number of separate rocket boosters used in a multistage launch. More boosters introduce more propellant to launch a vehicle further. For our experiments, multistage launches include 1-3 boost stages. Our dataset contains 1-boost, 2-boost, and 3-boost ground launches as well as air launches for each type of vehicle, resulting in a total of 207 trajectories. We have 146 HGV trajectories (85 HGV-1 trajectories, HGV-2 trajectories) and 61 CRV trajectories, as detailed in Table 6.1. Each HGV possesses a different aerodynamic

profile and thus exhibits distinct maneuvering capabilities. Our engagement planning would vary when considering each of these vehicles, so it is important to discriminate between HGV-1 and HGV-2 trajectories. The lengths of the trajectories vary since some vehicles and missions fly longer than others, as can be seen in Figure 6.1 and Figure 6.2.

Table 6.1. Vehicle Trajectory Dataset. This table provides details about the number of trajectories available per vehicle in the training, validation, and testing sets.

Vehicle Trajectory Dataset				
Vehicle	Train	Validation	Test	Total
HGV-1	59	8	18	85
HGV-2	42	6	13	61
CRV	42	6	13	61
Total	143	20	44	207

Next, we implement two different trajectory truncation methods to construct the final inputs to the ML methods. The first approach divides full trajectories into shorter segments of equal length using a rolling window method. The window commences at the beginning of a trajectory, corresponding to time after lift off (TALO) of zero seconds. Then, it traverses the entire trajectory, shifting by one second of data each time. Thus, trajectory segments in the dataset overlap. The length of the window determines the number of segments available for training, validation, and testing. In other words, a larger window will result in fewer segments because it will reach the end of a trajectory sooner than shorter window will. For the experiments in this thesis, windows of size 10, 20, 100, and 200 seconds were used.

The second approach divides trajectories into three equal parts corresponding to phase-1, phase-2, and phase-3 data. Because full trajectories are unequal in length, this approach results in segments of unequal length within each phase of flight. Furthermore, the segments do not overlap. An example of this trajectory division is depicted in Figure 6.3. Trajectories were divided in this way to create balanced datasets for each phase of flight. This approach ensures the same amount of data is available for training and evaluating each separate

model, as described later in this section. For both approaches, once the trajectories have been segmented, the neural network classifies the segments independently of each other.

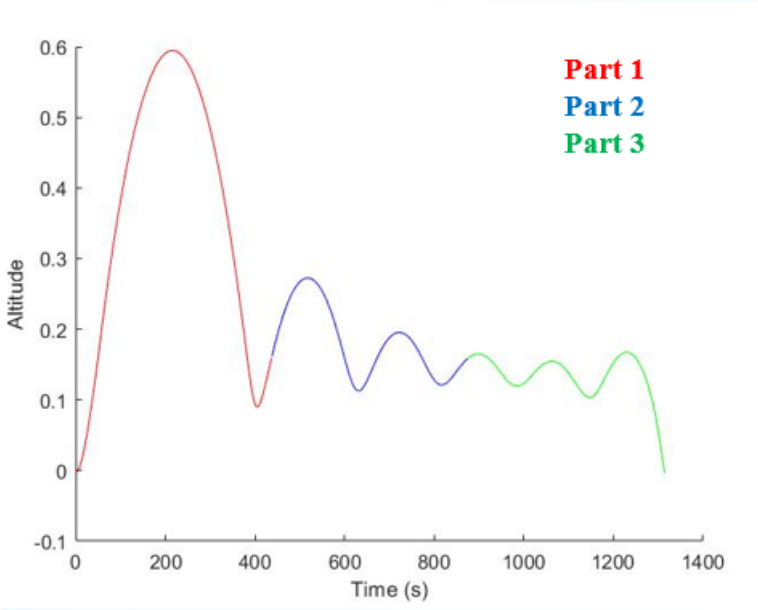


Figure 6.3. Trajectory Division into Phases. This plot depicts a trajectory divided into phases equal in length.

A set of seven datasets serve as the basis for these initial experiments. The details of each dataset are included in Tables 6.2, 6.3, 6.4, 6.5, 6.6, 6.7, and 6.8. For the creation of each dataset, trajectories of each vehicle class are separated into training, validation, and testing sets according to a 70%:15%:15% split. Then, data from the terminal phase of flight is excluded. In initial experiments, data from the boost phase of flight is also excluded. However, later experiments pivot and begin to include boost phase segments in the dataset. As a result, only Vehicle Segment Dataset 1 (Table 6.2) and Vehicle Segment Dataset 2 (Table 6.3) do not include boost phase data. Next, the trajectories are segmented via the rolling window approach. One final step for Vehicle Segment Dataset 2 (Table 6.3) involves randomly sampling HGV-1 segments to include a subset of them in the dataset. In general, more HGV-1 trajectories (and thus segments) are available in these datasets, so experiments conducted with Vehicle Segment Dataset 2 (Table 6.3) explore performance on a more balanced dataset.

Table 6.2. Vehicle Segment Dataset 1. This table provides details about the number of 100-second length segments (excluding boost phase data) available per vehicle in the training, validation, and testing sets.

Vehicle Trajectory Dataset 1				
Vehicle	Train	Validation	Test	Total
HGV-1	71,286	14,560	20,260	106,106
HGV-2	37,250	8,449	9,545	55,244
CRV	30,826	6,706	6,565	44,097
Total	139,362	29,715	36,370	205,447

Table 6.3. Vehicle Segment Dataset 2. This table provides details about the number of 100-second length segments (excluding boost phase data) available per vehicle in the training, validation, and testing sets. This is a more balanced dataset, with equal numbers of segments in the HGV-1 and HGV-2 classes.

Vehicle Trajectory Dataset 2				
Vehicle	Train	Validation	Test	Total
HGV-1	37,250	8,449	9,545	55,244
HGV-2	37,250	8,449	9,545	55,244
CRV	30,826	6,706	6,565	44,097
Total	105,326	23,604	25,655	154,585

6.3.2 Multi-Layer Perceptron (MLP) Methods

In this section we describe our MLP methods for vehicle classification. We explored four different network architectures, which are depicted in Figures 6.4 thru 6.7. Before evaluating data with the MLP-based approaches, the trajectory segments undergo pre-processing steps to construct the final inputs to the networks. Given 3-D position coordinates (*i.e.*, longitude, latitude, and altitude) of a vehicle, other aerodynamic features, such as speed, are calculated. Then, polynomials are fitted to each of these aerodynamic features, resulting in a set of 36 coefficients. The coefficients serve as the inputs to the MLPs.

Table 6.4. Vehicle Segment Dataset 3. This table provides details about the number of 10-second length segments (including boost phase data) available per vehicle in the training, validation, and testing sets.

Vehicle Trajectory Dataset 3				
Vehicle	Train	Validation	Test	Total
HGV-1	79,959	16,324	22,318	118,601
HGV-2	43,424	9,772	11,015	60,418
CRV	37,000	8,029	8,035	49,403
Total	160,383	34,125	41,368	228,422

Table 6.5. Vehicle Segment Dataset 4. This table provides details about the number of 20-second length segments (including boost phase data) available per vehicle in the training, validation, and testing sets.

Vehicle Trajectory Dataset 4				
Vehicle	Train	Validation	Test	Total
HGV-1	79,369	16,204	22,178	117,751
HGV-2	43,004	9,682	10,915	63,601
CRV	36,580	7,939	7,935	44,519
Total	158,953	33,825	33,093	225,871

The first architecture, MLP-max, employs three separate MLPs. Each one is trained for a different binary classification problem: *HGV-1 vs Not*, *HGV-2 vs Not*, and *CRV vs Not*. In each case, the *Not* category consists of data from the other two types of vehicles. For example, in the *HGV-1 vs Not* scenario, HGV-2 and CRV segments comprise the *Not* class. After training each of the MLPs, testing data is processed by each of them. Each MLP predicts whether the segment under analysis **S** is the vehicle type it was trained to identify or not. The output of each of the MLPs is a likelihood score in the range [-1, 1], where a "1" indicates that the network believes the segment to be of the vehicle type it was trained to identify with strong likelihood. On the other hand, a score of "-1" indicates that network believes **S** belongs to the *Not* category with strong likelihood. The vehicle class

Table 6.6. Vehicle Segment Dataset 5. This table provides details about the number of 100-second length segments (including boost phase data) available per vehicle in the training, validation, and testing sets.

Vehicle Trajectory Dataset 5				
Vehicle	Train	Validation	Test	Total
HGV-1	74,649	15,244	21,058	110,951
HGV-2	39,644	8,962	10,115	58,721
CRV	33,220	7,219	7,135	47,574
Total	147,513	31,425	38,308	217,246

Table 6.7. Vehicle Segment Dataset 6. This table provides details about the number of 200-second length segments (including boost phase data) available per vehicle in the training, validation, and testing sets.

Vehicle Trajectory Dataset 6				
Vehicle	Train	Validation	Test	Total
HGV-1	68,749	14,044	19,658	102,451
HGV-2	29,020	6,319	6,135	41,474
CRV	35,444	8,062	9,115	52,621
Total	133,213	28,425	34,908	196,546

corresponding to the maximum likelihood score of the three MLPs' outputs is the prediction of MLP-max.

The second architecture is named MLP-softmax. It utilizes only one MLP with a softmax layer in order to operate in a 3-class fashion. The output of this network consists of three probabilities in the range [0, 1] that indicate the chances that the segment under analysis **S** belongs to each of the vehicle classes. The vehicle class corresponding to the maximum probability is the prediction of MLP-softmax. This network architecture differs from MLP-max in that it is trained to learn how to differentiate between three different vehicles concurrently, whereas MLP-max requires three different networks trained to learn how to differentiate between only two different vehicles concurrently. In other words, no decision-making function is added to MLP-softmax after training.

Table 6.8. Vehicle Segment Dataset 7. This table provides details about the number of segments of various lengths (including boost phase data) available per vehicle in the training, validation, and testing sets.

Vehicle Trajectory Dataset 7				
Vehicle	Train	Validation	Test	Total
HGV-1	41	8	10	59
HGV-2	32	6	8	46
CRV	32	6	8	46
Total	105	20	26	151

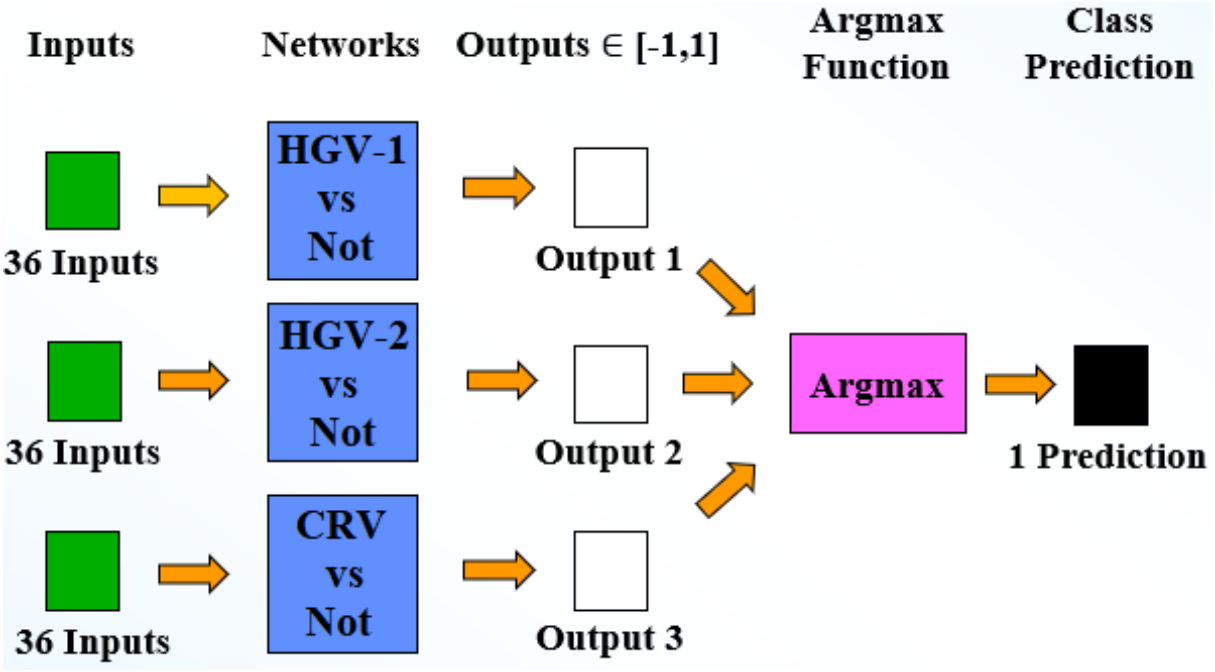


Figure 6.4. MLP-max. MLP-max consists of three MLPs trained in a 2-class fashion.

The third architecture, MLP-DC, is very similar to MLP-max. However, it differs in the way that the outputs of each separate MLP are evaluated to make a final prediction. Just as with MLP-max, each of the MLPs are trained separately on a different 2-class problem. For testing, though, a threshold function and a decision tree are added to evaluate the MLP outputs. The threshold function performs binary thresholding on the likelihood scores to convert them to either a "1" or a "-1." Next, the decision tree determines whether there

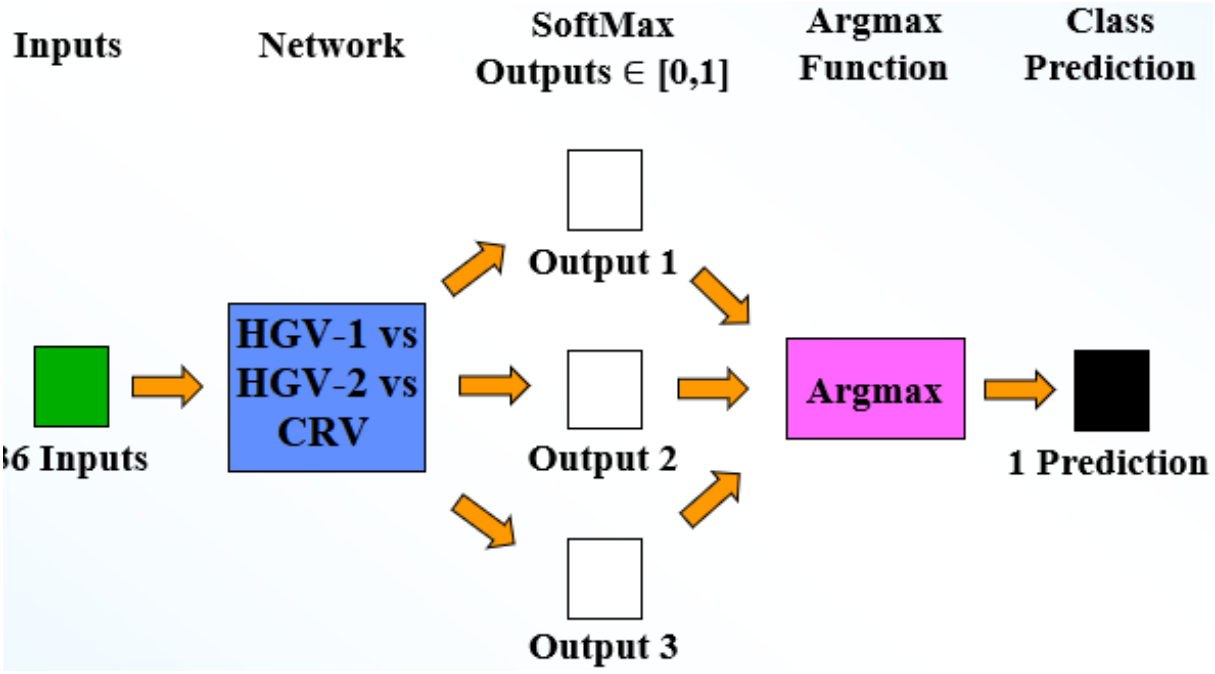


Figure 6.5. MLP-softmax. MLP-softmax consists of one MLP trained in a 3-class fashion.

is a clear classification choice or not. In other words, if there is only one "1" in the set of thresholded results, that vehicle class will be used as the final prediction. Thus, a threshold output of [1 -1 -1] corresponds to HGV-1, [-1 1 -1] corresponds to HGV-2, and [-1 -1 1] corresponds to CRV. All other combinations of result in a classification of *Not*. Thus, this network architecture operates in a 4-class fashion.

MLP-DC2 builds upon MLP-DC and utilizes a 2-layer decision tree. First, seven separate MLPs are trained to learn *HGV-1 vs Not*, *HGV-2 vs Not*, *CRV vs Not*, *HGV-1 vs HGV-2*, *HGV-2 vs CRV*, *CRV vs HGV-1*, and *HGV-1 vs HGV-2 vs CRV*. Then, once training is complete, threshold functions and a 2-layer decision three are added to connect and evaluate the outputs. In the first layer of decision making, the outputs of *HGV-1 vs Not*, *HGV-2 vs Not*, and *CRV vs Not* are evaluated. Binary thresholding on this set of outputs is performed. If there is a clear classification choice, meaning that the thresholded set of values only contains one "1", that vehicle class is used as the final prediction. However, if there is any uncertainty, the MLP corresponding to the classes where the uncertainty lies is used to evaluate **S** further. For example, if the thresholded set of values contains a "1" corresponding

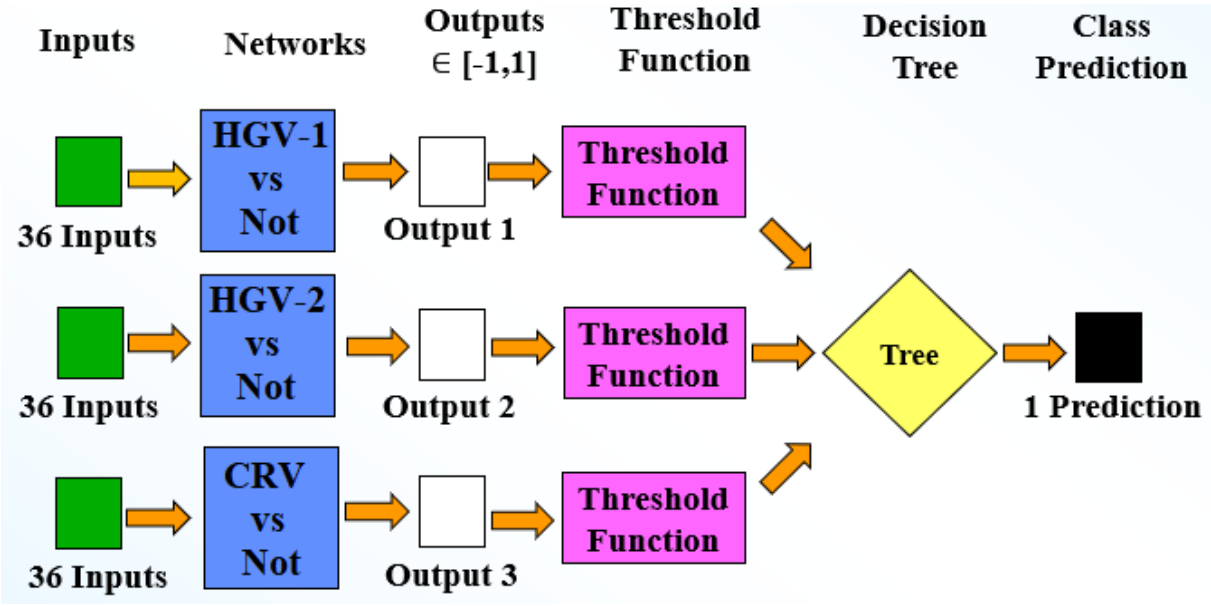


Figure 6.6. MLP-DC. MLP-DC consists of three MLPs and a 1-layer decision tree. Each MLP is trained in a 2-class fashion.

to both the HGV-1 and HGV-2 classes, the *HGV-1 vs HGV-2* network will be utilized; if the thresholded set of values contains all "1"s, the *HGV-1 vs HGV-2 vs CRV* network will be utilized; etc. **S** is processed by the next network, and then either another binary thresholding (in the case of a 2-class network) or a softmax (in the case of a 3-class network) function is utilized to achieve the final prediction.

6.3.3 Long Short-Term Memory (LSTM) Methods

The LSTM-1 architecture pivots from MLPs to LSTMs. It employs one LSTM trained in a 3-class scenario to concurrently learn to classify segments into three vehicle categories. The network produces three probabilities corresponding to each of the vehicle classes, so a softmax function is once more utilized to obtain the final prediction.

Finally, the last set of experiments utilizes three separate LSTMs (LSTM-2-1, LSTM-2-2, LSTM-2-3) trained on a separate phase of flight (phase-1. phase-2, phase-3). Each LSTM functions in a 3-class fashion, so softmax is again used to evaluate the outputs of each LSTM and determine the final prediction.

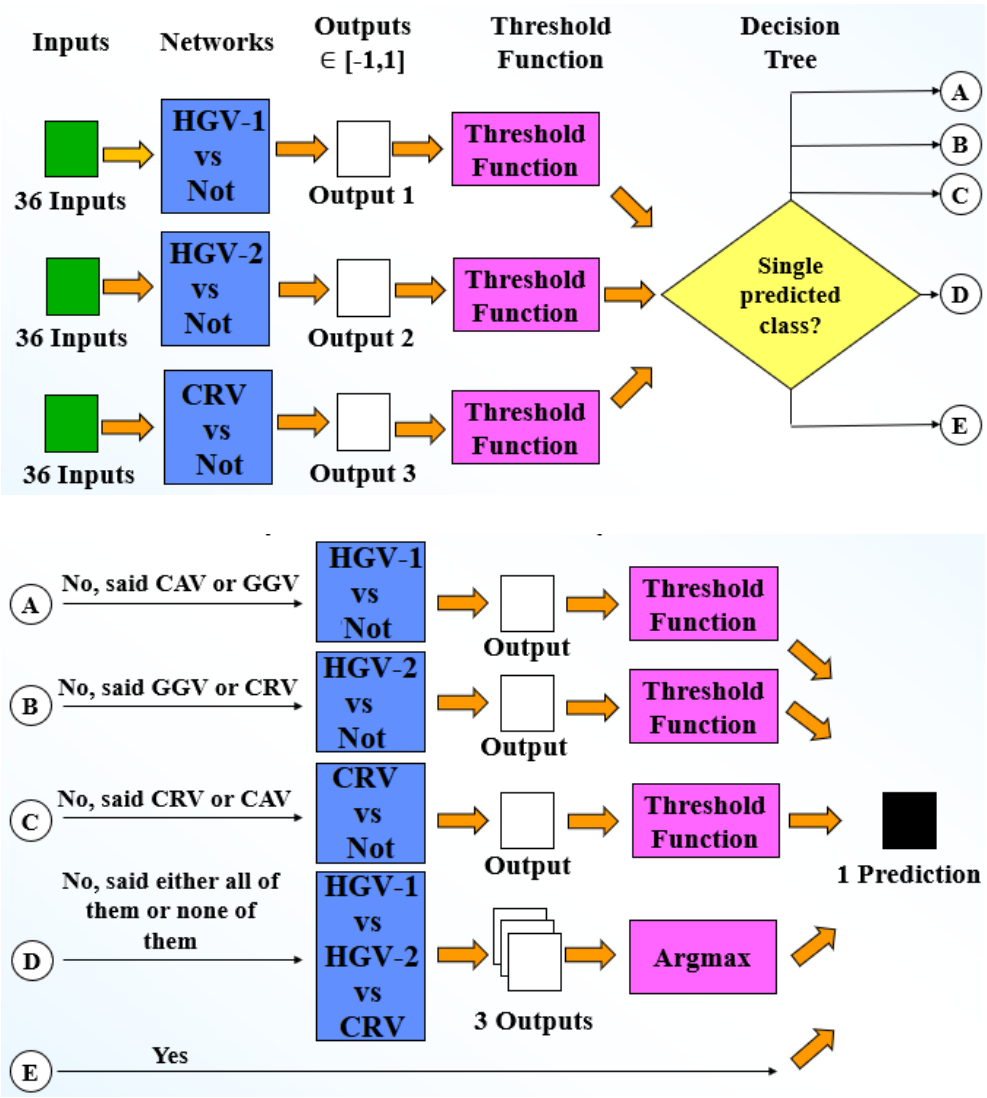


Figure 6.7. MLP-DC2. MLP-DC2 consists of seven MLPs and a 2-layer decision tree. Six MLPs are trained in a 2-class fashion, and the remaining one is trained in a 3-class fashion.

6.3.4 Initial Experimental Results

In this section, we present experimental results using MLP and LSTM approaches. Table 6.9 summarizes the results of several experiments conducted with different datasets, architectures, and data features. The first column in the table specifies the architecture (or specific model within an architecture) used in an experiment. The second column establishes the dataset utilized for the set of experiments. The third column includes the data features

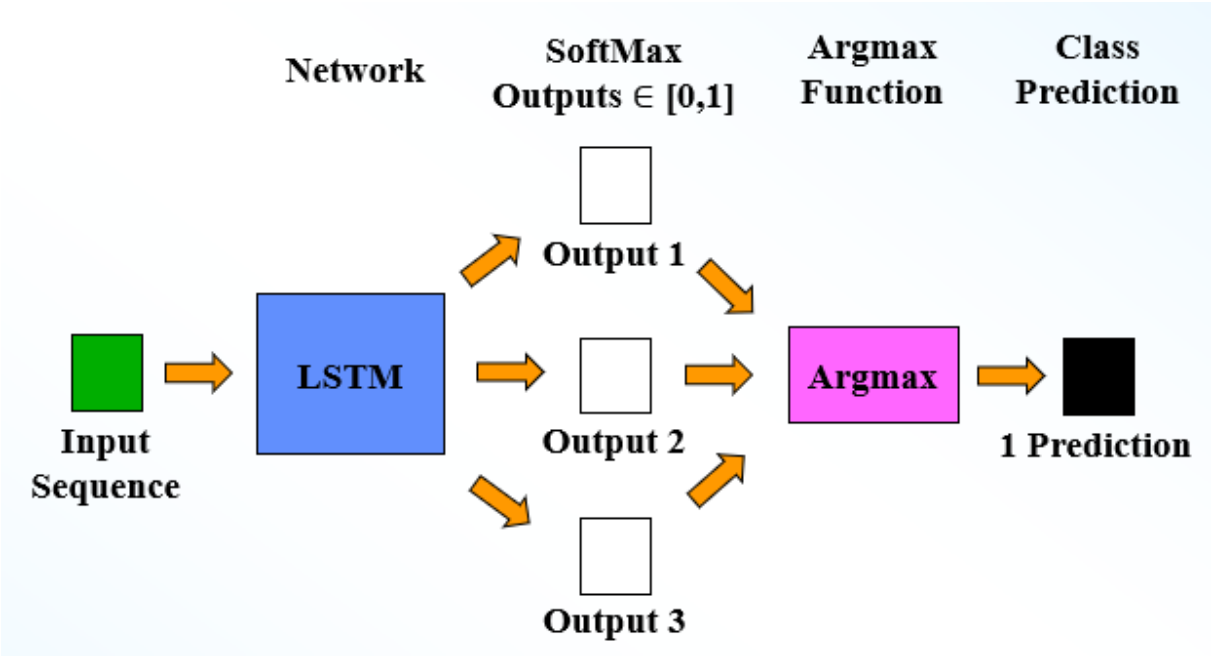


Figure 6.8. LSTM-1. LSTM-1 consists of one LSTM trained in a 3-class fashion.

used as inputs to the the network: T refers to normalized TALO with respect to each individual trajectory, A refers to altitude, L1 refers to longitude, and L2 refers to latitude. The next three columns report average training, validation, and testing accuracies, when available, of the experiments conducted. Experiments administered with MLP-based architectures were repeated five times, and experiments using LSTM-based architectures were repeated ten times.

Experiments conducted with MLP-based architectures utilize Vehicle Segment Dataset 1 or Vehicle Segment Dataset 2, which contain segments of 100 seconds. MLP-max performs the best of all architectures in these experiments, achieving an average testing accuracy of 58.30%. However, the results of all these MLP experiments stagnate, not achieving better results with different configurations. For this reason, we commenced use of LSTMs.

Experiments conducted with LSTM-1 utilize datasets of varying lengths of segments to explore the impact of segment length on performance. Some of the experiments use normalized TALO as an input feature as well. For experiments both with TALO as an input and without, performance on Vehicle Segment Dataset 5 of 100-second segments achieves the best performance. Using TALO as an input feature with this setup yields an average

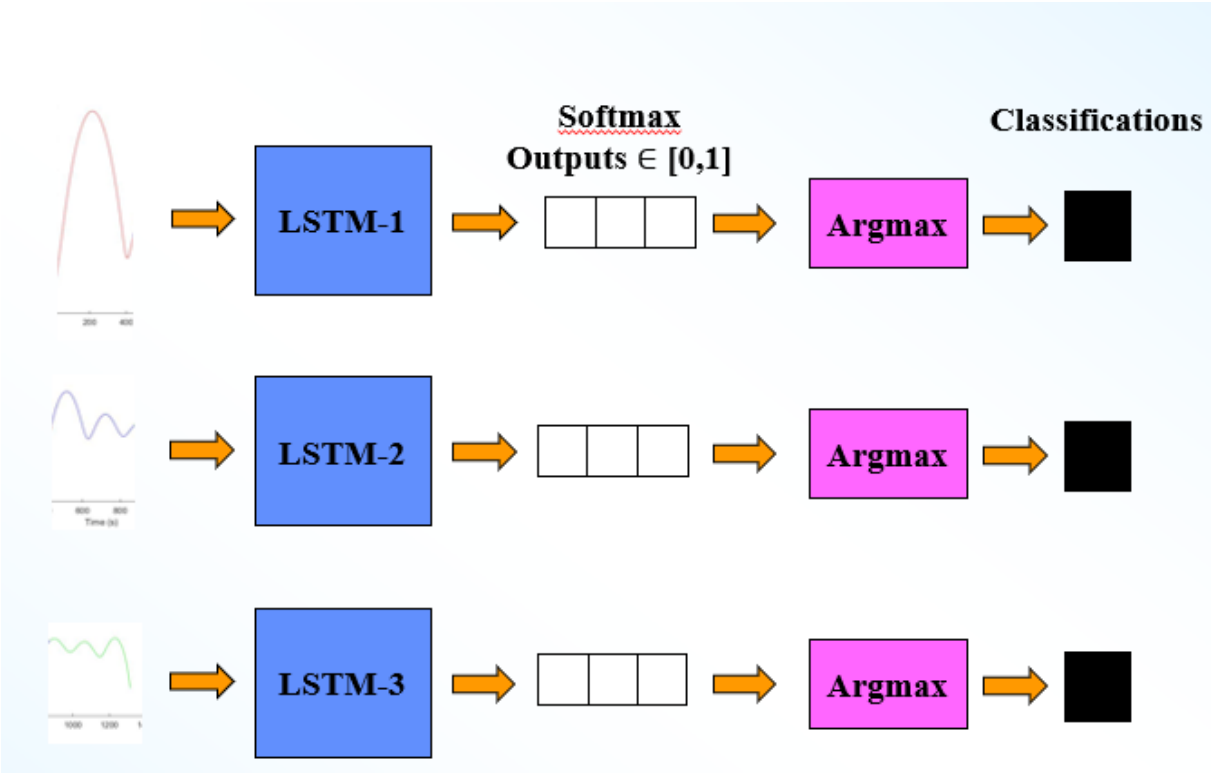


Figure 6.9. LSTM-2. This configuration consists of three LSTMs trained in a 3-class fashion. Each LSTM is trained on a different set of phase data.

testing accuracy of 97.73%, which is the best overall performance. In general, adding TALO as an input feature increased average performance of the network. No trends in performance related to segment length can be identified with certainty, though.

The final set of experiments analyze each phase of a trajectory. Training these models proved very challenging, and many experimental setups and model parameter changes were conducted before LSTM-2-1 showed signs of learning. Unfortunately, learning was never achieved with LSTM-2-2 or LSTM-2-3. Both of these networks will only predict HGV-1, and thus the performances reported in Table 6.9 are the percentage of HGV-1 data samples in Vehicle Segment Dataset 7. One reason these networks did not learn as well (or at all) as previous architectures is that much less data is available for training these networks. Compared to Vehicle Segment Dataset 3 that contains 228,000 segments, Vehicle Segment Dataset 7 only contains 151. Another challenge for this architecture is the longer input sequences. The average segment lengths by vehicle for Vehicle Segment Dataset 7 is 588.78

Table 6.9. Initial Results. Results of initial experiments conducted with MLPs and LSTMs.

Initial Vehicle Classification Results					
Architecture	Dataset	Features	Average Training Accuracy	Average Validation Accuracy	Average Testing Accuracy
MLP-max	1	A, L1, L2	-	-	58.30%
MLP-max	2	A, L1, L2	-	-	49.65%
MLP-softmax	1	A, L1, L2	-	-	54.34%
MLP-softmax	2	A, L1, L2	-	-	41.08%
MLP-DC	1	A, L1, L2	-	-	40.91%
MLP-DC2	1	A, L1, L2	-	-	56.60%
LSTM-1	3	A, L1, L2	82.67%	82.30%	84.22%
LSTM-1	4	A, L1, L2	87.70%	87.65%	88.07%
LSTM-1	5	A, L1, L2	90.45%	90.18%	91.70%
LSTM-1	6	A, L1, L2	72.92%	72.82%	76.36%
LSTM-1	3	T, A, L1, L2	90.80%	90.60%	91.63%
LSTM-1	4	T, A, L1, L2	84.98%	84.37%	86.22%
LSTM-1	5	T, A, L1, L2	97.18%	97.21%	97.73%
LSTM-1	6	T, A, L1, L2	83.42%	83.29%	85.25%
LSTM-2-1	7	T, A, L1, L2	65.05%	68.67%	61.54%
LSTM-2-2	7	T, A, L1, L2	39.05%	40.00%	38.46%
LSTM-2-3	7	T, A, L1, L2	39.05%	40.00%	38.46%

seconds for HGV-1s, 407.65 seconds for HGV-2s, and 417.41 seconds for CRVs. These are significantly longer inputs than the datasets consisting of segments 10, 20, 100, and 200 seconds in length.

6.3.5 Discussion of Initial Work

The proposed technique exploits a data driven approach, thus learning how to distinguish HGV-1s from HGV2-s from CRVs directly from the available training data. Results show

that the developed methodology accomplishes classification to varying degrees of accuracy on the used datasets. While some networks, such as LSTM-2-2 never learn to classify vehicle flight paths, other networks, such as LSTM-1, achieve promising results. Although the results of LSTM-1 experiments are very good, more investigation into explaining these results is necessary. Furthermore, it would be interesting to see how the technique performs on segments of even shorter lengths to test how little data is needed to classify a trajectory segment. Finally, we will develop other methods to pursue the goal of determining how quickly a trajectory can be classified.

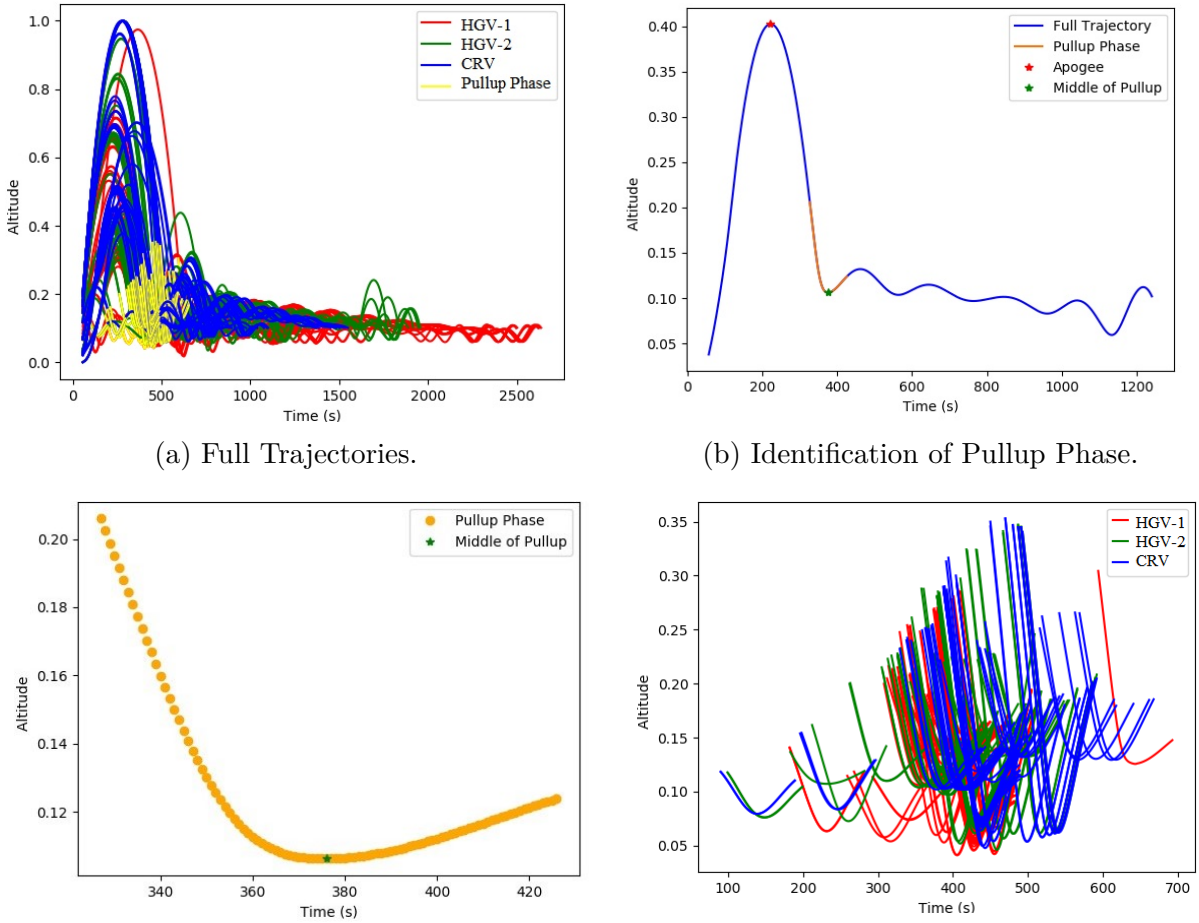
One criticism of these approaches is that it is difficult to justify or explain what the ML methods learn from the data. Although LSTM-1 performed well on virtually all datasets and segment lengths, it is difficult to understand why this network achieves such high performance. Classifying 10-seconds of data along any point in a trajectory that exhibits maneuvers and different flight events is a challenge, yet LSTM-1 achieved very good performance with Vehicle Segment Dataset 3. Without physically enforcing the models to learn how a vehicle flies from position data, it is difficult to explain why they can succeed with such information. Aerodynamic domain knowledge suggests that this task is not trivial either. Instead, other aerodynamic features, such as angle of attack, angle of attack rate, and dynamic pressure might yield to better and more explainable discrimination results.

Even though some experimental results show promising results, it is necessary to better define a problem in which the ML models' decisions can be better understood. Due to the uncertainty in what the methods learn with this experimental setup, we propose a different problem formulation that analyzes new aerodynamic features. The next section presents the new methodology, which serves as our current state-of-the-art experimental setup and validation.

6.4 Proposed Approach

Given trajectory data, we seek to identify the vehicle that created the trajectory. We evaluate our approach on a dataset containing trajectories of three different types of vehicles. Two vehicles are HGVs, which we will refer to as HGV-1 and HGV-2. The third vehicle is

a conic re-entry vehicle (CRV). We investigate the characterization performance of three different machine learning models, including a SVM, a KNN, and a CNN) [89]–[91], [153], [154].



(a) Full Trajectories.

(b) Identification of Pullup Phase.

(c) Pullup Phase of One Trajectory.

(d) Pullup Phases of all Trajectories.

Figure 6.10. Extracting Pullup Phases. These four plots show the process of extracting the pullup phase from all of the trajectories in the dataset.

6.4.1 Extracting Aerodynamic Features of Pullup Phases

Our approach analyzes the pullup phase from a vehicle's mission. Figure 6.10a shows all of the trajectories in the dataset and their pullup phases. The pullup phases, indicated in yellow, start at different times, depending on the characteristics of the vehicle and the mission. We use altitude measurements of a trajectory to identify the pullup phase. We

identify the apogee of a trajectory, indicated by a red star in Figure 6.10b. The apogee is the highest point of a trajectory and occurs early in a vehicle's flight. After reaching apogee, the vehicle dives into the remainder of the flight and flies at lower altitudes. It uses its momentum and potential energy harvested from its decline to glide to its final destination. Next, we identify the first local minima in the trajectory after the apogee. This point is indicated by a green star in Figure 6.10b. The local minima results from the vehicle's dive from apogee and pullup from the dive into the next maneuver of the mission.

Then, we obtain an equal number of measurements from before and after the pullup point. We extract the same number of time samples from all of the trajectories to ensure that the learning system evaluates uniform-length inputs. Figure 6.10b indicates the pullup phase in orange. For our experiments, we consider the pullup phase to be 100 seconds long and centered around the first local minima. Figure 6.10c is a closer look at the pullup phase described by 100 seconds of altitude data. We observe that the altitude data points before the middle of the pullup phase are spaced further apart than the points after the pullup phase. Since our data contains evenly spaced measurements (in terms of time), this reveals that the vehicle travels at a faster velocity in its dive right before it pulls up. We use this pre-processing for all trajectories to identify the times associated with their pullup phases. Figure 6.10d shows the altitude measurements of the pullup phases from all of the trajectories in the dataset.

After the times of the pullup phases are identified, we estimate the aerodynamic features. We consider the aerodynamic features of angle of attack, angle of attack rate, and dynamic pressure, which are shown in Figure 6.11 for the full trajectories. Dynamic pressure is derived from estimated track kinematic states. Angle of attack and rate are augmented filter states. Figure 6.12 shows these aerodynamic features for the pullup phases. The segments in Figure 6.12 serve as the inputs to the machine learning models in our experiments. These indicate that there is a high degree of overlap between the pullup phases of different vehicles.

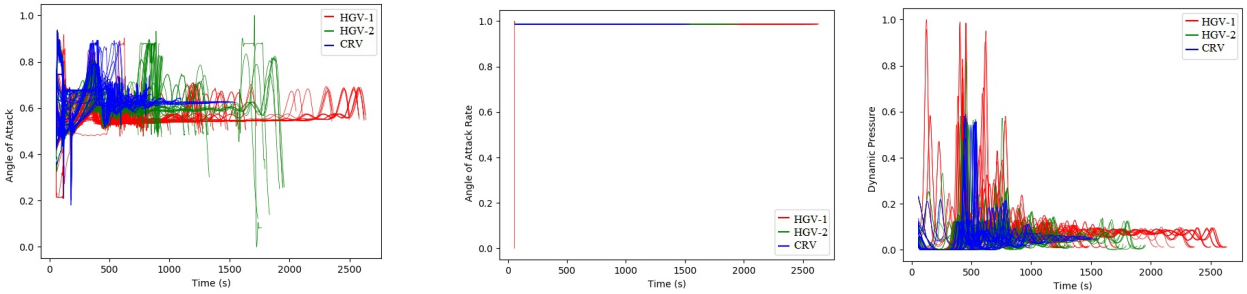


Figure 6.11. Full Trajectories. Aerodynamic features of full trajectories before desired region is extracted.

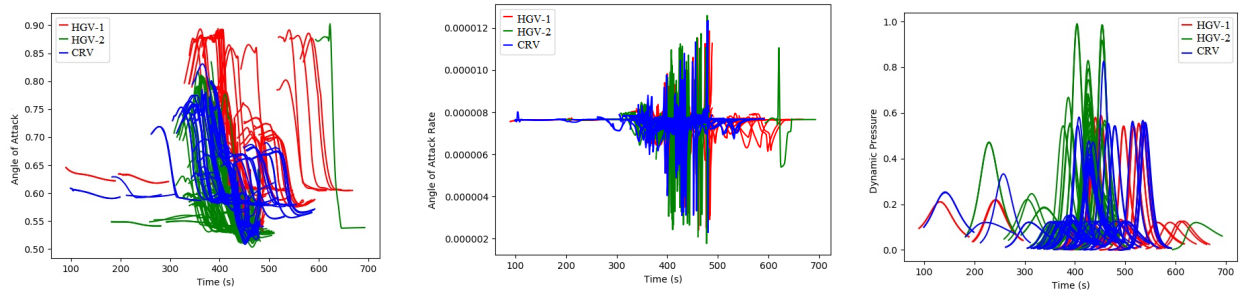


Figure 6.12. 100-Second Pullup Phases. Aerodynamic features of 100-Second pullup phases classified with Machine Learning methods.

6.4.2 Noisy Aerodynamic Features

We also examine the situation with noisy observations. The segments shown in Figure 6.12 are pristine (noiseless) trajectories. We need to determine the effect of noise on our classification task. To simulate this we add zero-mean Gaussian-distributed noise to each aerodynamic feature based on the feature's range $(max - min)$ for a particular pullup phase. For an aerodynamic feature f of trajectory t, $f_{t,p,max}$ is the maximum value of the feature associated with the pullup phase p for t. Likewise, $f_{t,p,min}$ is the minimum value of the aerodynamic feature associated with the pullup phase p for trajectory t. Let M be the "scale factor"of the noise which controls the standard deviation. The standard deviation of the Gaussian noise added to feature f for trajectory t is:

$$\sigma_{t,p,f} = (f_{t,p,max} - f_{t,p,min}) * M. \tag{6.1}$$

Thus, different noise signals will be added to each pullup phase based on the aerodynamic features for that particular trajectory. Figures 6.13, 6.14, and 6.15 show aerodynamic features with noise scale factor $M = 0.05$. Figures 6.16, 6.17, and 6.18 show aerodynamic features with noise scale factor $M = 0.25$. In our experiments we used noise scale factors $M \in \{0, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50\}$.

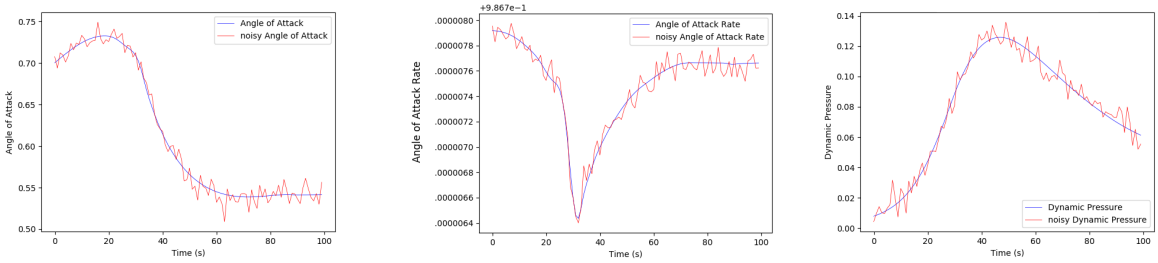


Figure 6.13. HGV-1 with Small Noise Level. Aerodynamic features for HGV-1 with $M = 0.05$.

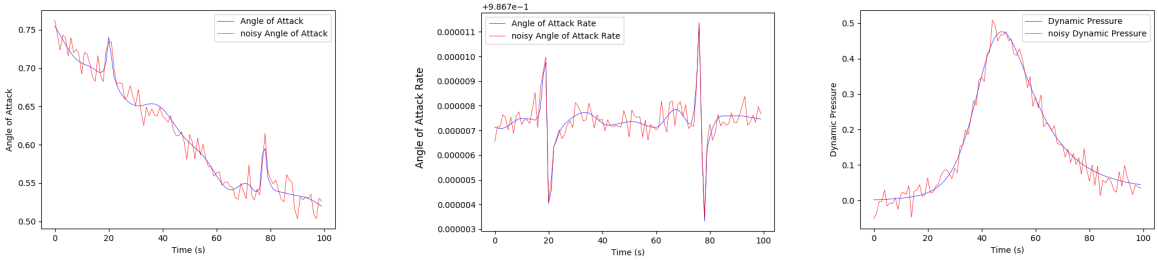


Figure 6.14. HGV-2 with Small Noise Level. Aerodynamic features for HGV-2 with $M = 0.05$.

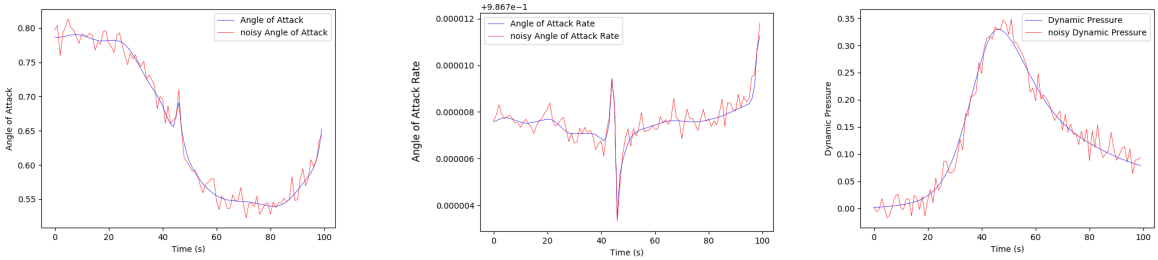


Figure 6.15. CRV with Small Noise Level. Aerodynamic features for CRV with $M = 0.05$.

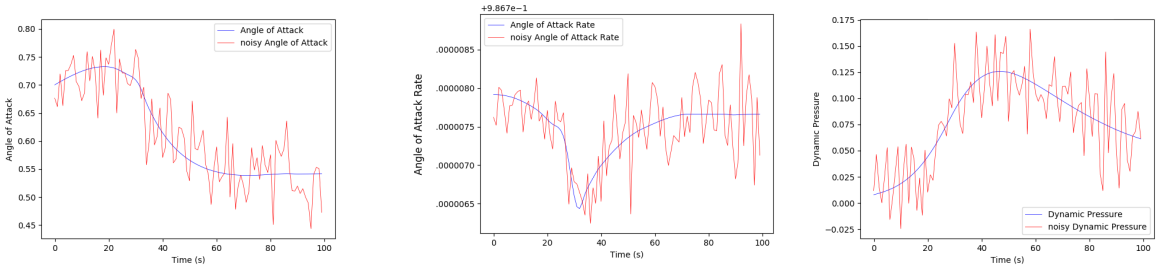


Figure 6.16. HGV-1 with Medium Noise Level. Aerodynamic features for HGV-1 with $M = 0.25$.

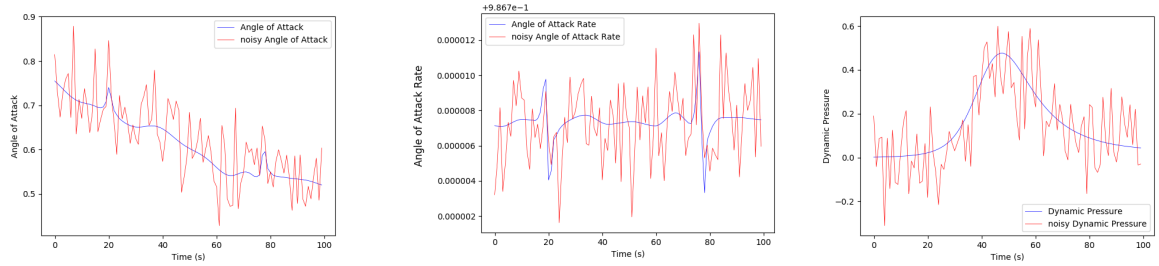


Figure 6.17. HGV-2 with Medium Noise Level. Aerodynamic features for HGV-2 with $M = 0.25$.

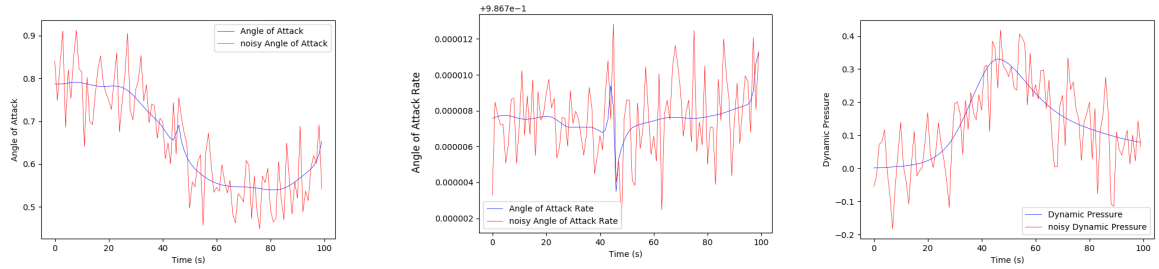


Figure 6.18. CRV with Medium Noise Level. Aerodynamic features for CRV with $M = 0.25$.

6.4.3 K-Nearest Neighbors (KNN)

A machine learning method we investigate is KNN [89], [90]. KNN does not require any prior knowledge about the underlying probability density function of the dataset. This non-parametric method strives to classify data samples based on proximity of a trajectory under analysis (*i.e.*, trajectories in the testing set) to all other known trajectories (*i.e.*, trajectories in the training set). For our experiments, we classify a trajectory based on the $k = 1$ nearest neighbor to that trajectory, where proximity is determined by the L_2-norm. To form the

inputs to the KNN, we row concatenate each of the 1-D aerodynamic features. Since we are analyzing 3, 100-second-long aerodynamic features, the input to the KNN for one trajectory will be a row vector with length 300.

6.4.4 Support Vector Machine (SVM)

We also investigate a Support Vector Machine (SVM) with a Radial Basis Function (RBF) kernel [91]. Because our classification task contains more than two classes, we train three different SVMs on three different binary classification tasks. Each SVM uses a one-vs-one approach, meaning that it learns to differentiate between two different vehicle classes. This is different from the one-vs-many approach, which learns to differentiate between one class and the rest of the vehicles in the dataset. In order to choose a final classification for a trajectory, the argmax function is employed on the three output scores from each of the SVMs. We utilize the same row concatenation procedure that we use for the KNN classifier to create the inputs to the SVM.

6.4.5 Convolutional Neural Network (CNN)

The final machine learning method we investigate is a CNN [153], [154]. Our proposed CNN architecture is shown in Figure 6.19, and the details about the sizes of the layers are presented in Table 6.10. The neural network consists mainly of two 1-D convolutional layers to analyze the 1-D aerodynamic features presented to it. It also employs max pooling and dropout layers to regularize the model and prevent overfitting. The final output of the neural network uses a softmax function to a fully-connected dense layer consisting of three nodes. This step produces three detection scores, indicating the likelihood that a trajectory under analysis belongs to each of the three classes. Next, the argmax function is used to these detection scores, producing a final class prediction. During training, we use early stopping with a patience of 5 epochs and the Adam optimizer [78]. CNNs can operate on 2-D matrices, so we present temporally aligned aerodynamic features to the CNN for analysis. A single trajectory input to the CNN has dimensions 3 x 100.

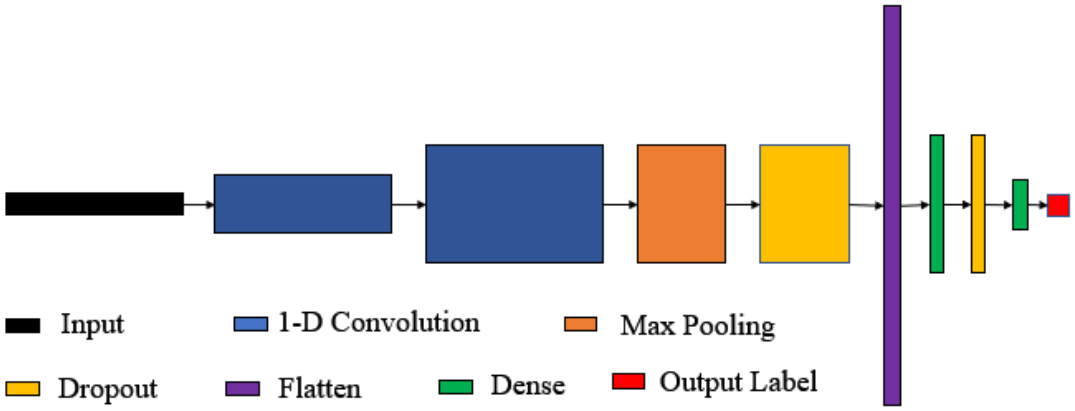


Figure 6.19. CNN Diagram for Vehicle Classification. The CNN developed for our vehicle classification approach.

6.5 Experimental Setup

6.5.1 Dataset

To create our experimental dataset, we simulate estimated angle of attack, angle of attack rate, and dynamic pressure at a 1 Hz sampling rate for entire trajectories, excluding boost and terminal phases. Angle-of-attack and angle-of-attack rate can be estimated from lift-over-drag ratio and velocity, given knowledge about the threat aerodynamic profile. The trajectories and associated features are normalized in order to enable better learning of the machine learning models [139]–[142]. Our dataset contains trajectories resulting from various launch configurations. Each launch configuration is defined by its start location and rocket stages. For our experiments, start positions include the ground and the air. In the case of an air launch, vehicles are released from another aircraft that already achieved a desire altitude. The number of rocket stages refers to the number of separate rocket boosters used in a multistage launch. More boosters introduce more propellant to launch a vehicle further. For our experiments, multistage launches include 1-3 boost stages. Our dataset contains 1-boost, 2-boost, and 3-boost ground launches as well as air launches for each type of vehicle, resulting in a total of 207 trajectories. We have 146 HGV trajectories (85 HGV-1 trajectories,

Table 6.10. CNN Architecture for Vehicle Classification. This table indicates the architecture of the proposed CNN. Each row in the table specifies *(from left to right)* the layer type, its output shape, and the number of parameters per layer. The outputs of the CNN can be described in the form (L, N). L is the length of the 1-D aerodynamic parameter vector of measurements. N is the number of feature maps produced.

CNN Architecture		
Layer	Output Shape (L, N)	Parameters
input	(100, 3)	0
conv_1	(98, 32)	128
conv_2	(96, 64)	6,208
max pooling	(48, 64)	0
dropout_1	(48, 64)	0
flatten_1	(3072, 1)	0
dense_1	(128, 1)	393,344
dropout_2	(128, 1)	0
dense_2	(3, 1)	387
output label	(1,1)	0

HGV-2 trajectories) and 61 CRV trajectories. Each HGV possesses a different aerodynamic profile and thus exhibits distinct maneuvering capabilities. Our engagement planning would vary when considering each of these vehicles, so it is important to discriminate between HGV-1 and HGV-2 trajectories. The lengths of the trajectories vary since some vehicles and missions fly longer than others, as can be seen in Figure 6.1 and Figure 6.2.

We train our machine learning methods on a subset of the trajectories of the full dataset, referred to as the training set. For the CNN, we determine when training finishes by monitoring the error of the CNN on another subset of the data. This is the validation set. We evaluate the performance of the CNN on the validation set each time that the CNN analyzes all trajectories in the training set. Once the error of the CNN stops decreasing significantly on the validation set, we conclude that the CNN is no longer effectively learning from the training data and terminate any further training. Note that SVMs and KNNs have set training times based on the size of the dataset, so the validation set is not used with either of

these methods. Finally, once a method has been trained, we evaluate the trained model by examining how it performs on the maneuvers on the remaining trajectories in the dataset, which will be referred to as the testing set. We partition the full dataset into three sets for training, validation, and testing according to a 70%:10%:20% split. Details about the dataset appear in Table 6.1. Table 6.11 is a replica of Table 6.1, reproduced here for reference.

Table 6.11. Vehicle Trajectory Dataset. This table provides details about the number of trajectories available per vehicle in the training, validation, and testing sets.

Vehicle Trajectory Dataset				
Vehicle	Train	Validation	Test	Total
HGV-1	59	8	18	85
HGV-2	42	6	13	61
CRV	42	6	13	61
Total	143	20	44	207

6.5.2 Training and Evaluation Process

For our experiments, we train each of our machine learning methods multiple times independently. Each time a new method is trained, a new dataset split is used. This means that a trajectory might be included in the training set to train one version of a method. Then, it might be included in the testing set another time when a new version of the method is trained. We utilize this random split because we have a relatively small dataset. Rather than splitting the dataset once and using that split for all experiments, we train our methods on different trajectories each time and see how it performs on a new set of trajectories. To ensure we account for the impact of different train/validation/test splits, we train 500 versions of each method and evaluate their average performance results. To validate our approach, we evaluate all machine learning methods based on accuracy, precision, recall, and F1-score metrics. Furthermore, we analyze how classification accuracy relates to TALO.

6.5.3 Heuristic Methods Used in Evaluation

We establish two theoretical bounds on performance using heuristic classifiers. The heuristic methods serve as minimum viable benchmarks for improvement. These are important to consider in an unbalanced dataset, such as ours. If, for example, our model performs worse than random guessing, it is a poor classifier. A more challenging benchmark to beat is a classifier that predicts the majority class in the dataset. If our model performs worse than predicting the most common vehicle, it is of no use at all. Our method must outperform these baselines in order to demonstrate effectiveness.

The first heuristic baseline method is a theoretical classifier that performs random guessing. For an unbalanced, 3-class dataset, the accuracy of a random classifier A_R is calculated by:

$$A_R = \sum_{v \in V} P(T = v) * P(C = v) \approx 0.3419, \tag{6.2}$$

where $V = \{HGV - 1, HGV - 2, CRV\}$, $P(T = v)$ refers to the probability that a trajectory T is of vehicle type v, and $P(L = v)$ refers to the probability that the classifier predicts a class C of vehicle v. This theoretical classifier will be referred to as Baseline-Rand.

The second theoretical classifier we consider predicts trajectories as belonging to the mode of the dataset. The mode is the majority class, which is the vehicle that appears most often in the dataset. In our case, the classifier would predict HGV-1 every time it analyzes a trajectory. Thus, it would obtain an accuracy of $\frac{18}{44} * 100\% \approx 40.91\%$. This benchmark method is often called the 0-R (*i.e.*, zero rule) [155], [156]. It represents the accuracy of a classifier that predicts the most frequently occurring class in an unbalanced dataset. We will refer to it as Baseline-0R.

6.5.4 Results on Noiseless Data

The first row in Table 6.12 summarizes the average results achieved by training and evaluating 500 models independently on noiseless data (*i.e.*, $M = 0$) for each method. Results indicate that the CNN and the KNN perform very well. Both the CNN and the KNN results

significantly outperform the two heuristic benchmarks as well as the SVM. The SVM does not even exceed the heuristic benchmarks for all metrics considered. It fails to surpass both the random guessing benchmark (*i.e.*, Baseline-Rand) and the 0-R benchmark (*i.e.*, Baseline-0R) with its F1-score, indicating that this method performs extremely poorly on this classification task. The CNN improves in accuracy by more than a factor of two over the Baseline-0R benchmark. Furthermore, the CNN increases classification accuracy by ∼43% in comparison to the SVM. To verify that our method performs well even on an unbalanced dataset, we consider the F1-score. The CNN also achieves a higher F1-score compared to both benchmarks and outperforms the SVM by ∼74%. The KNN classifier performs the best of all the methods. It achieves the highest accuracy, precision, recall, and F1-score of all machine learning methods. In fact, it achieves a margin of less than 4% to perfection based on all metrics.

6.5.5 Results on Noisy Data

Next, we explore the performance of our methods on noisy data. The rest of Tables 6.12 and 6.13 summarize the results of all methods trained and tested on noisy data with the same magnitude M. In other words, a model is trained on noisy data with $M = 0.15$ and then tested on data with $M = 0.15$. Then, it is trained on noisy data with $M = 0.20$ and tested on noisy data with $M = 0.20$. Tables 6.14 and 6.15 show results of training all methods on noisy data and testing them on noiseless data. For this scenario, noise with magnitude M was used both the training and validation sets, while the testing set remains noiseless. Tables 6.16 and 6.17 show results for the opposite scenario, where training and validation sets remain noiseless, while noise with magnitude M is applied to the testing data. For all of these scenarios, we train 500 separate instances of each method for each noise level and test them independently.

The SVM performs poorly for all noisy scenarios. In most cases, it achieves F1-scores that are worse than random guessing. There are a few noisy scenarios for which the SVM achieves F1-scores that are on par with the Baseline-Rand benchmark, but these cases are infrequent. Interestingly, its performance rather consistently fails for all metrics in that its

performance does not decline as noise levels increase. For all scenarios and all noise levels, it achieves roughly the same performance. Figure 6.20 and Figure 6.21 show these results more explicitly.

The CNN achieves better results for noisy scenarios. It performs much better than the SVM, Benchmark-Rand, and Benchmark-0R for all noise levels and noise scenarios. Only the KNN outperforms it when $M < 0.25$. It also achieves fairly consistent results in this range despite increasing noise levels. Figures 6.20 and Figure 6.21 visualize the consistent behavior of the CNN on noisy data from $M = 0$ until $M = 0.3$. Even as the noise level increases on both the training and testing sets, the CNN identifies vehicles well. At $M = 0.3$, it still achieves $\sim$74% accuracy and an F1-score of $\sim$71%. Once $M > 0.25$, the CNN outperforms all other methods, including the KNN. Its classification accuracy and F1-scores decrease more as noise levels increase, but the CNN continues to identify vehicles better than any other methods with high levels of noise..

It might seem surprising that the CNN performs well in spite of noise. Injecting noise into data analyzed by a neural network is a form of data augmentation that enhances the network's capabilities [157]–[159]. This technique, often called "adding jitter", introduces more regularization effect into the network. As a result, it decreases the likelihood of overfitting and increases a model's generalization capabilities. It is especially important to utilize data augmentation techniques for relatively small datasets, such as ours. Small datasets provide fewer data samples and capture less of the input space. Thus, it is easier for a neural network to memorize the dataset rather than learning a proper mapping from the input space to the desired output space. Adding noise to the inputs provides a smoother, richer representation of the input space and enables a model to learn an appropriate mapping from data to labels. Introducing noise to our dataset enables better learning for our CNN and increases the model's robustness.

The KNN achieves the highest classification metrics overall. For $M < 0.10$, the KNN classifier identifies vehicles with greater than 90% accuracy, precision, recall, and F1-score. It also surpasses the CNN (and all other methods) for half of the noise levels we consider. However, the KNN experiences the greatest degradation in performance as noise level increases and noise scenarios change. Its classification accuracy and F1-score decrease more

rapidly than any other method for the scenario when training and testing data have noise. When $M > 0.25$, the KNN success drops to below that of the CNN. When $M > 0.40$, the KNN accuracies drop below those of the SVM, and the KNN F1-scores indicate worse performance than the Baseline-0R heuristic. The KNN F1-score ranges from 96.76% when $M = 0$ to 53.08% when $M = 0.50$, resulting in a decrease in performance of ∼45%. On the other hand, the CNN only experiences a ∼22% decrease in F1-score over this range. This is true for the scenario where the KNN is trained on data with noise and tested on pristine data as well. However, the KNN holds up well when trained on noiseless data and tested on noisy data. In this scenario, its performance degradation is much more gradual, for this is only a change in F1-score of ∼9% from $M = 0$ to $M = 0.50$, while the CNN experiences a ∼14% decrease in F1-score over this range. Even at $M = 0.30$, it achieves accuracy, precision, recall, and F1-score greater than 93.5%.

6.5.6 Discussion

These results demonstrate that our methods successfully differentiate vehicles by analyzing key aerodynamic features. Since our CNN and KNN achieve the highest performances of all noise levels and scenarios, we consider other factors about each of these methods in addition to their performances. For example, KNN is more sensitive to noise. If a method needs to operate in a situation with higher noise levels, it is preferable to use the CNN. The CNN performs well fairly consistently despite various noise levels and noise scenarios.

Another important factor to consider is the size the dataset. A KNN classifier must compare a trajectory under analysis to all data points already labeled and available in the training set. Thus, as the size of the training set increases, so will the time complexity of the KNN analysis. At some point, the inference time (*i.e.*, the time required to analyze a single, new trajectory) will exceed the acceptable amount of time required to analyze a new data sample. The CNN has a very quick inference time that does not scale with the size of the training set. The CNN can be trained in an offline scenario and used in an online scenario to analyze new trajectories quickly. There is more work required to determine the appropriate amount of time for which to train a CNN, but once it is trained properly, it will

execute quickly. Finally, the translational equivariant properties of CNNs lend themselves to analyzing trajectory data with respect to time.

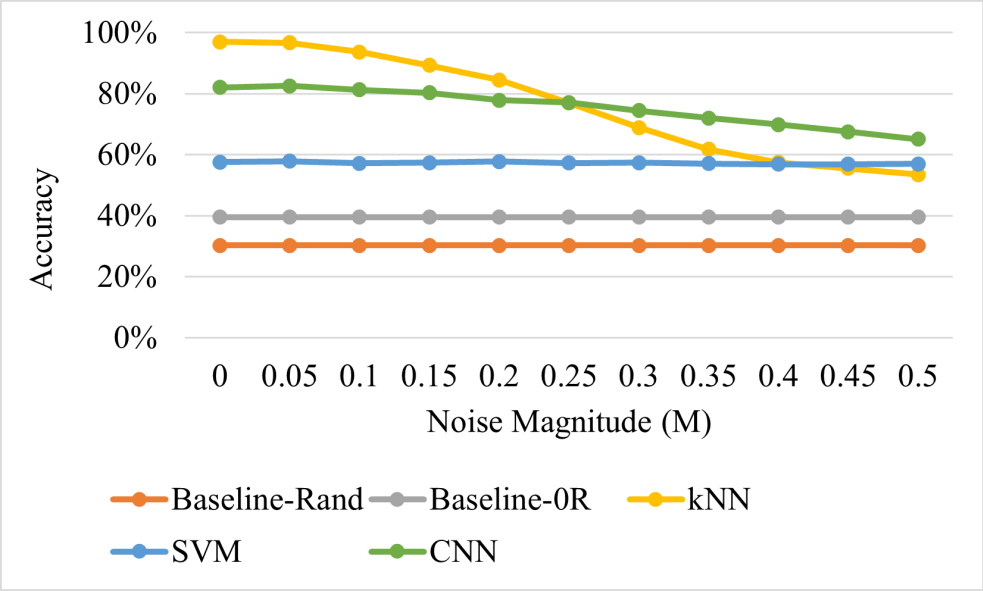


Figure 6.20. Vehicle Classification Average Accuracy Versus Noise Magnitude. Results of 500 experimental runs for each machine learning method with various noise magnitudes applied to training, validation, and testing sets.

6.5.7 Results with Respect to Time After Lift-Off (TALO)

Figure 6.22 and Figure 6.23 provide another analysis of the CNN's performance on noisy data. These plots show how classification accuracy behaves as a function of time after lift off (TALO), both from a summary perspective and as a breakdown based on each vehicle type. To generate these results, we use a growing window approach. To start, the window has a width of one second. A trained CNN classifies every pullup phase in the dataset based on its first second of data. Next, the window size increases to two seconds of data, and the first two seconds of every trajectory are classified. The process continues until the window size is equal to the length of the pullup phase, and every single second of data has been classified.

We see that initially, classification results are ~40% when $M = 0.20$ and $M = 0.25$. By analyzing the corresponding accuracy by vehicle type, we see that HGV-1 is classified with

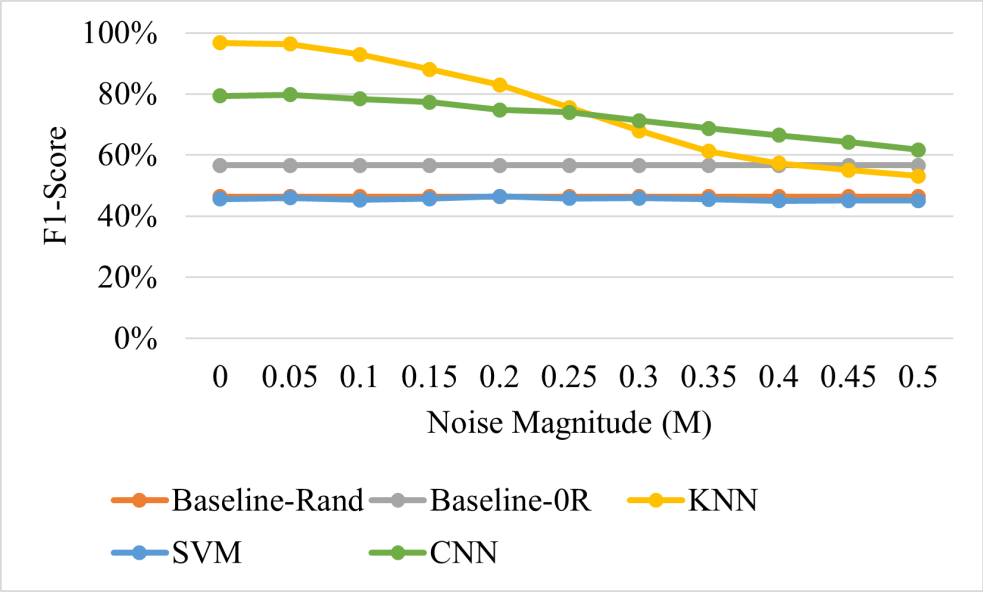


Figure 6.21. Vehicle Classification Average F1-Score Versus Noise Magnitude. Results of 500 experimental runs for each machine learning method with various noise magnitudes applied to training, validation, and testing sets.

100% accuracy initially. This indicates that these trained CNNs began by predicting HGV-1 every single time they analyze a new trajectory. This initial stage behaves the same as our Baseline-0R classifier. As more data becomes available for analysis and reaches the original input length to the CNN (*i.e.*, 100 seconds), classification results increase. This reveals that the equivariant properties of the CNN enable classification of trajectories when a trajectory under analysis is incomplete or unaligned with the pullup phase. The CNN does not need to train on data from full trajectories in order to achieve high classification. Furthermore, it achieves high accuracy on the critical initial phases of flight.

Table 6.12. Results with Noise Magnitudes in Range 0-0.25 Applied to All Data Splits. This table contains the average results for each metric of 500 experimental runs on noisy data, where equal noise was applied to training, validation, and testing sets.

Results with Noise in Range 0-0.25 Applied to All Data Splits					
Noise Magnitude	Method	Average Accuracy	Average Precision	Average Recall	Average F1
0.00	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	57.56%	47.25%	52.28%	45.58%
	CNN	82.10%	83.49%	80.70%	79.36%
	KNN	97.02%	97.03%	96.78%	96.76%
0.05	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	57.84%	48.44%	52.59%	45.97%
	CNN	82.55%	83.92%	81.12%	79.80%
	KNN	96.65%	96.55%	96.36%	96.32%
0.10	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	57.15%	47.13%	51.84%	45.27%
	CNN	81.25%	82.70%	79.67%	78.36%
	KNN	93.69%	93.47%	93.03%	92.96%
0.15	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	57.37%	48.99%	52.09%	45.75%
	CNN	80.28%	81.54%	78.65%	77.27%
	KNN	89.27%	88.86%	88.20%	88.04%
0.20	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	57.81%	50.72%	52.58%	46.45%
	CNN	77.84%	79.12%	76.20%	74.77%
	KNN	84.49%	84.01%	83.30%	82.96%
0.25	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	57.26%	49.51%	51.98%	45.83%
	CNN	77.06%	78.28%	75.33%	74.02%
	KNN	77.00%	76.92%	76.01%	75.60%

Table 6.13. Results with Noise Magnitudes in Range 0.3-0.5 Applied to All Data Splits. This table contains the average results for each metric of 500 experimental runs on noisy data, where equal noise was applied to training, validation, and testing sets.

Results with Noise in Range 0.3-0.5 Applied to All Data Splits					
Noise Magnitude	Method	Average Accuracy	Average Precision	Average Recall	Average F1
0.30	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	57.34%	50.26%	52.07%	45.85%
	CNN	74.45%	75.66%	72.69%	71.32%
	KNN	68.84%	70.19%	68.11%	67.92%
0.35	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	57.10%	49.57%	51.81%	45.54%
	CNN	71.99%	73.32%	70.03%	68.76%
	KNN	61.74%	64.77%	61.27%	61.22%
0.40	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	56.88%	48.56%	51.57%	45.07%
	CNN	69.84%	70.59%	67.89%	66.50%
	KNN	57.51%	61.92%	57.36%	57.28%
0.45	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	56.87%	49.16%	51.58%	45.16%
	CNN	67.56%	68.58%	65.70%	64.22%
	KNN	55.55%	60.10%	55.33%	55.09%
0.50	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	56.95%	48.27%	51.69%	45.11%
	CNN	65.03%	66.24%	63.28%	61.74%
	KNN	53.47%	58.54%	53.36%	53.08%

Table 6.14. Results with Noise Magnitudes in Range 0-0.25 Applied to Training Data Splits. This table contains the average results for each metric of 500 experimental runs on noisy data, where equal noise was applied to training and validation sets.

Results with Noise in Range 0-0.25 Applied to Training Data Splits					
Noise Magnitude	Method	Average Accuracy	Average Precision	Average Recall	Average F1
0.00	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	57.56%	47.25%	52.28%	45.58%
	CNN	82.10%	83.49%	80.70%	79.36%
	KNN	97.02%	97.03%	96.78%	96.76%
0.05	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	57.92%	48.40%	52.70%	46.00%
	CNN	82.31%	83.55%	80.83%	79.44%
	KNN	96.98%	96.90%	96.73%	96.68%
0.10	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	58.11%	49.78%	52.91%	46.62%
	CNN	81.96%	82.90%	80.52%	79.28%
	KNN	95.22%	95.14%	94.76%	94.68%
0.15	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	58.11%	49.78%	52.91%	46.62%
	CNN	80.92%	82.45%	79.37%	77.93%
	KNN	91.69%	91.80%	90.81%	90.60%
0.20	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	56.96%	48.82%	51.65%	45.43%
	CNN	80.40%	82.25%	78.68%	77.28%
	KNN	87.26%	87.70%	86.03%	85.54%
0.25	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	57.13%	49.49%	51.82%	45.60%
	CNN	78.61%	80.58%	76.98%	75.44%
	KNN	81.31%	81.77%	80.32%	79.44%

Table 6.15. Results with Noise Magnitudes in Range 0.3-0.5 Applied to Training Data Splits. This table contains the average results for each metric of 500 experimental runs on noisy data, where equal noise was applied to training and validation sets.

Results with Noise in Range 0.3-0.5 Applied to Training Data Splits					
Noise Magnitude	Method	Average Accuracy	Average Precision	Average Recall	Average F1
0.30	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	57.46%	51.15%	52.20%	46.16%
	CNN	76.35%	78.03%	74.55%	72.62%
	KNN	68.41%	70.12%	67.52%	67.03%
0.35	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	57.50%	50.82%	52.24%	46.24%
	CNN	74.17%	76.84%	72.07%	70.12%
	KNN	60.90%	64.76%	60.58%	60.03%
0.40	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	56.74%	49.39%	51.41%	45.24%
	CNN	71.12%	74.39%	69.08%	66.82%
	KNN	55.66%	62.25%	55.79%	55.29%
0.45	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	56.48%	48.54%	51.11%	44.75%
	CNN	68.81%	72.93%	66.70%	64.32%
	KNN	53.57%	61.31%	54.04%	53.24%
0.50	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	56.42%	49.07%	51.03%	44.79%
	CNN	66.75%	71.23%	65.08%	62.27%
	KNN	52.35%	60.46%	52.83%	51.94%

Table 6.16. Results with Noise Magnitudes in Range 0-0.25 Applied to Testing Split. This table contains the average results for each metric of 500 experimental runs on noisy data, where noise was applied to only the testing set.

Results with Noise in Range 0-0.25 Applied to Testing Data Split					
Noise Magnitude	Method	Average Accuracy	Average Precision	Average Recall	Average F1
0.00	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	57.56%	47.25%	52.28%	45.58%
	CNN	82.10%	83.49%	80.70%	79.36%
	KNN	97.02%	97.03%	96.78%	96.76%
0.05	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	57.57%	47.40%	52.28%	45.58%
	CNN	82.60%	83.92%	81.08%	79.85%
	KNN	96.91%	96.87%	96.68%	96.64%
0.10	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	57.73%	47.43%	52.46%	45.68%
	CNN	82.30%	83.50%	80.80%	79.62%
	KNN	96.78%	96.73%	96.51%	96.47%
0.15	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	57.07%	46.73%	51.74%	44.90%
	CNN	80.86%	82.19%	79.36%	78.17%
	KNN	96.63%	96.58%	96.35%	96.32%
0.20	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	57.51%	48.00%	52.21%	45.47%
	CNN	80.50%	81.47%	79.08%	77.99%
	KNN	95.99%	95.92%	95.66%	95.62%
0.15	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	57.19%	47.99%	51.86%	45.27%
	CNN	78.47%	79.66%	76.84%	75.60%
	KNN	95.15%	95.05%	94.71%	94.66%

Table 6.17. Results with Noise Magnitudes in Range 0.3-0.5 Applied to Testing Data Split. This table contains the average results for each metric of 500 experimental runs on noisy data, where noise was applied to only the testing set.

Results with Noise in Range 0.3-0.5 Applied to Testing Data Split					
Noise Magnitude	Method	Average Accuracy	Average Precision	Average Recall	Average F1
0.30	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	57.24%	48.11%	51.98%	45.36%
	CNN	77.27%	78.25%	75.73%	74.68%
	KNN	94.17%	94.01%	93.60%	93.56%
0.35	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	56.88%	46.97%	51.52%	44.67%
	CNN	75.60%	76.50%	74.04%	72.81%
	KNN	93.12%	92.89%	92.48%	92.41%
0.40	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	56.95%	47.18%	51.60%	44.73%
	CNN	73.67%	74.58%	72.00%	70.74%
	KNN	91.69%	91.39%	90.89%	90.82%
0.45	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	56.63%	47.73%	51.27%	44.53%
	CNN	73.03%	74.34%	71.44%	70.36%
	KNN	90.59%	90.31%	89.77%	89.66%
0.50	Baseline-Rand	34.19%	100.00%	34.19%	46.43%
	Baseline-0R	40.91%	100.00%	40.91%	56.66%
	SVM	56.22%	46.77%	50.81%	44.05%
	CNN	71.19%	72.48%	69.58%	68.54%
	KNN	89.20%	88.80%	88.29%	88.18%

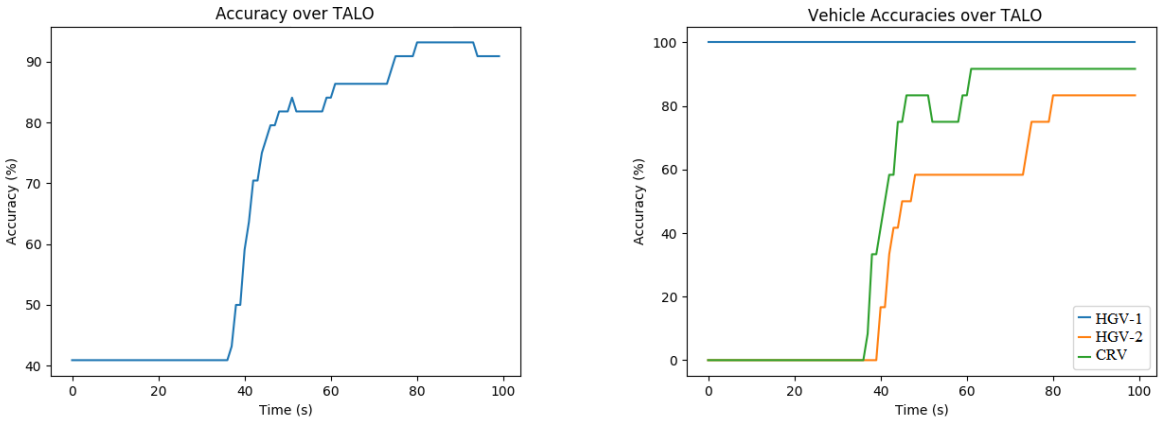


Figure 6.22. Vehicle Classification Average Accuracy over TALO with $M = 0.20$ for a Single Trained CNN. These TALO plots show average accuracy achieved on all vehicles in the testing dataset using one of the 500 trained CNNs on data containing noise with $M = 0.20$. The noise is applied to the training, validation, and testing sets.

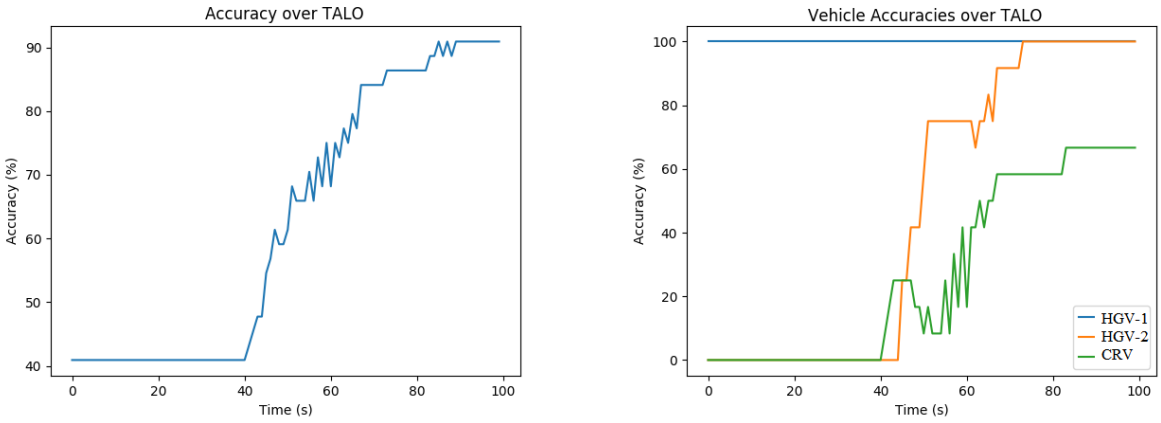


Figure 6.23. Vehicle Classification Average Accuracy over TALO with $M = 0.25$ for a Single Trained CNN. These TALO plots show average accuracy achieved on all vehicles in the testing dataset using one of the 500 trained CNNs on data containing noise with $M = 0.25$. The noise is applied to the training, validation, and testing sets.

7. MACHINE LEARNING FOR HYPERSONIC VEHICLE PREDICTION

7.1 Overview

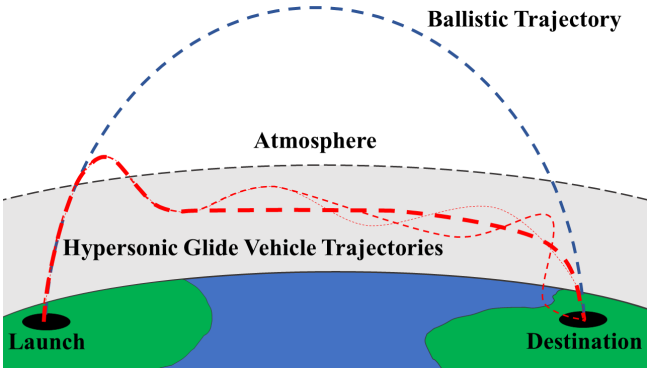


Figure 7.1. Different Vehicle Trajectory Comparisons. Examples of hypersonic glide vehicle (HGV) trajectories (shown in red) and a ballistic trajectory (shown in blue). HGVs can follow many different flight paths to reach a specific destination.

Hypersonic glide vehicles (HGVs) fly at high velocities with a high degree of maneuverability. More specifically, they fly at velocities faster than Mach 5 (*i.e.*, five times the velocity of sound) [54], [160]–[162]. This combination of speed and maneuverability differentiates them from traditional spacecraft, which cannot achieve such velocities and agility at the same time. For example, spacecraft (such as those used in the Apollo missions [163]) and ballistic vehicles (such as those proposed for commercial travel by SpaceX in 2017 [164]–[166]) accelerate to hypersonic velocities for short periods of time as they re-enter the Earth's atmosphere [161], [163], [167]. Ballistic vehicles follow parabolic trajectories (known as ballistic trajectories) from their initial launch points to their final destinations [168]–[172], as shown in Figure 7.1. Their trajectories extend high out of Earth's atmosphere to sub-orbital space, so they must re-enter the atmosphere during descent. During re-entry, spacecraft and ballistic vehicles maintain velocities between Mach 5 and Mach 25 [54], [162], [163], [173]. However, they neither sustain these velocities for a long period of time nor maneuver efficiently [161], [167]. Hypersonic glide vehicles distinguish themselves by excelling at both speed and maneuverability – simultaneously.

HGVs travel at sustained hypersonic velocities for long distances by gliding [134]–[136], [138], [167]. They fly at lower altitudes than ballistic vehicles (as shown in Figure 7.1), which allows them to bounce off of the Earth's atmosphere. Each bounce increases their potential energy, which they use to "glide" further. Hypersonic glide vehicles steer themselves as they approach their landing spots, which enables them to approach from multiple directions; maneuver around bad weather conditions; and evade tracking systems [135], [138], [174]. Even en route, HGVs can change their destinations [135]. Figure 7.1 shows three examples of HGV trajectories. Although all three trajectories commence and end at the same locations, they maneuver to pursue different flight paths. For these reasons, predicting a HGV's flight behavior (*e.g.*, aerodynamics, flight path, maneuvers) is challenging.

In this chapter, we propose an approach to predict future flight phases of HGVs based on partially observed trajectories. Flight phases could be modeled as categorical labels that discretize HGV flight behavior. We define flight phases based on the vehicle's magnitude of rate of change of total energy. Total energy is the summation of kinetic and potential energy, estimated from a vehicle's altitude and 3-D velocity (*i.e.*, a subset of the vehicle's kinematic features). We use methods from NLP to model the flight phases as "words" and the HGV trajectories as "sentences." We learn a "grammar" from the HGV trajectories, which we use for our prediction task. Given "words" from the initial part of a HGV trajectory and the "grammar", we can predict future "words" in the "sentence" (*i.e.*, future HGV flight phases in the trajectory).

7.2 Related Work

Hypersonic glide vehicles present a new set of challenges for predicting long-term flight behavior (*e.g.*, aerodynamics, flight paths, maneuvers) because they exhibit such unprecedented capabilities. For example, analytical methods cannot exploit assumptions about Keplerian orbit (as ballistic vehicle prediction methods do [175]–[177]) to predict HGV trajectory shapes because HGVs exhibit complex, non-Keplerian behavior [168], [169]. Lei *et al.* [178] use a Kalman filter [179] for HGV trajectory prediction. First, a dynamics model is used to estimate future states (*e.g.*, position, velocity) of the vehicle. Then, as observa-

tions of the vehicle (*i.e.*, measurements of the vehicle in real-time) become available, the observations are used to update the state estimates using a Bayesian probabilistic approach. Accurate state estimates require accurate dynamics models and frequent measurements of a vehicle under analysis for Kalman filter methods to succeed [172], [179]. If measurements are not received often enough, then the state estimates could have poor accuracy or even result in filter divergence. However, real-time observations might not always be available to assist a Kalman filter in updating its dynamics model. Consequently, the Kalman filter might incorrectly estimate future states of the HGV [178], [180].

Some Kalman filter methods utilize an aerodynamic model that requires knowledge of a vehicle's aerodynamic profile. An aerodynamic profile of a vehicle controls how it flies, and different types of vehicles have different aerodynamic profiles. In reality, knowledge of features indicative of a vehicle's aerodynamic profile could be uncertain. Thus, a Kalman filter with an aerodynamic model can be difficult to implement. Other methods attempt to model HGV trajectories with polynomials [181], but this approach again requires many assumptions and knowledge for initialization. Some other research efforts use a conditional random field (CRF) [182] and a neural network [183] to predict hypersonic trajectories. The deep learning methods require extensive amounts of training data, though. For example, Xie *et al.* use 6,000 trajectories to train and validate their deep learning neural network [183]. In this work, we focus on the scenario where fewer training trajectories are known, and we assume that only initial portions of a trajectory inform our predictions. Our approach is also different in that we predict flight phases (rather than state estimates or flight paths). Later, we will provide the definition of flight phases for HGVs and explain how our proposed flight phases indicate possible flight maneuvers of HGVs.

We model HGV trajectories with a stochastic grammar [184]–[186]. Stochastic grammars originated from work in computational linguistics to formalize grammar theory and understand structures found in languages [184]–[186]. They deconstruct sequences into easily interpreted, hierarchical representations. Initially, they were used for language tasks, such as predicting ends of sentences given initial words in the sentences. Since then, they have been used for many other sequential data problems, such as predicting RNA structures [187]–[190]; forecasting human activities [191]–[193]; modeling human poses [194]; and

detecting anomalies in trajectory patterns [195]. Methods to derive a stochastic grammar typically do not require large training datasets [196].

Grammar models have been used previously for vehicle trajectory analysis [195], [197], [198]. In [195], Fanaswala *et al.* use a stochastic grammar to detect abnormal behavior (*e.g.*, circling around a building) of different vehicle trajectories (*e.g.*, car, ship, airplane). They approach this abnormal behavior detection task as a trajectory shape classification task, where certain trajectory shapes are defined as abnormal. First, quantized moving directions of the vehicles are used to build multiple grammars – one for each trajectory shape (*e.g.*, line, arc, rectangle, closed, move-stop-move). Then, a query trajectory and the grammars are used to estimate probabilities that indicate if the trajectory belongs to each grammar. The query trajectory is classified as the shape that corresponds to the most probable grammar, and the trajectory is considered abnormal if the shape is defined as abnormal. Brisaboa *et al.* use a grammar for trajectory compression [197]. They derive a grammar that describes flight patterns (based on a vehicle's position) in ship, commercial airline, and city taxi trajectories. They use the grammar to turn the trajectories (containing position coordinates) into "sentences" to compress the trajectories to 4% – 7% of their original sizes. López-Leonés *et al.* use a grammar to represent mathematical constraints for trajectory optimization, planning, and prediction [198]. They define a set of constraints that trajectories must satisfy. Then, they map the constraints to an "alphabet", where each "letter" represents a different constraint. The "letters" can be used to form "words" and a "grammar" to represent the combination of constraints. Trajectory prediction and planning methods utilize the grammar to interface with an Air Traffic Management (ATM) system that understands the grammar and how it relates to the flight constraints. In this way, the grammar is used to plan flight paths, ensure vehicles fly safely, and avoid collision in the airspace.

Stochastic grammar models consist of a set of production rules that define permissible sentence structures as well as probabilities of the rules occurring [184]–[186]. The format of the production rules depends on the type of grammar used. In our approach, we utilize a Probabilistic Context-Free Grammar (PCFG) [199]. Training a PCFG model requires labeled sequences. In NLP, the sequences could be sentences, and the smallest labeled units

of the sequences could be words. In our HGV task, the "words" are the flight phases, and the "sentences" are HGV trajectories represented by their magnitudes of rate of change of total energy. Figure 7.2 shows an example of a trajectory's magnitude of rate of change of total energy, which is used to determine the HGV's flight phases.

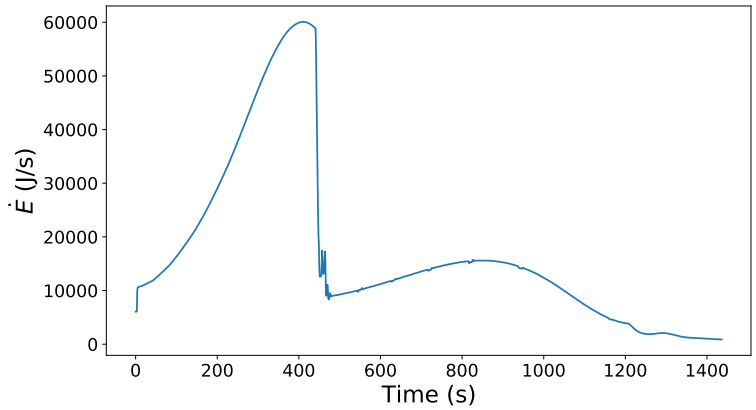


Figure 7.2. A HGV Trajectory. This plot shows a HGV trajectory based on its magnitude of rate of change of total energy (denoted as $\dot{E}$).

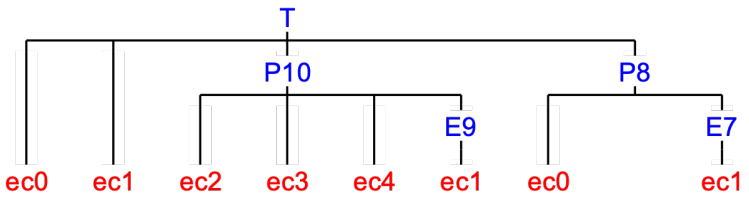


Figure 7.3. A HGV Trajectory as a Parse Graph. The parse graph represents a trajectory as a "sentence", where nodes and edges of the parse graph show the sentence structure of the trajectory. Flight phase labels (*i.e.*, "words") are shown in red; groups of flight phase labels (*i.e.*, higher-level "phrases" in a sentence) are shown in blue; and flight phase transition patterns (*i.e.*, production rules in a grammar) are shown as edges connecting the nodes in the parse graph.

We can illustrate the production rules (from the PCFG [199]) that a sequence exhibits with a tree known as a parse graph [200]. In a language application, a parse graph shows a hierarchical representation of a sentence and indicates more explicitly how each part of the sentence (*e.g.*, noun phrase, verb phrase, noun, verb) relates to the other words and phrases (*i.e.*, groups of words) in the sentence. In our approach, the parse graphs show trajectories

and how their flight phases relate. Figure 7.3 shows the same trajectory shown in Figure 7.2 formatted as a parse graph. Parse graphs decompose a trajectory into smaller components (*i.e.*, flight phases or groups of flight phases) based on the production rules. Each subsequent layer of the parse graph (proceeding from top to bottom) further divides a trajectory. Nodes located at the top of a parse graph represent longer portions of a trajectory, while nodes located towards the bottom represent shorter portions of a trajectory. Nodes at very bottom of the parse graph represent the smallest unit of the trajectory (*i.e.*, flight phases, which serve as "words" in the trajectories). Parse graphs also incorporate temporal information because they preserve the chronological order of flight phases. Time increases from left to right in a parse graph, where the left-most node ($ec0$ in Figure 7.3) shows a trajectory's earliest state, and the right-most node ($ec1$ in Figure 7.3) shows a trajectory's final state. Later in the chapter, we will provide more context about $ec0$, $ec1$, and the other nodes shown in Figure 7.3. The edges connecting the nodes in the parse graph represent the production rules in the PCFG (*i.e.*, transitions between different flight phases of a trajectory).

We combine two additional pieces of information with the PCFG to predict future flight phases. First, we identify the type of mission of a HGV trajectory. Some HGV missions fly longer distances than others. Long-range HGV missions exhibit different flight phases than short-range HGV missions, so we use a Long Short-Term Memory network [81] to identify the mission type (*i.e.*, short-range or long-range mission) of a trajectory to aid our prediction method. Second, we determine flight phase duration estimates to determine how long each flight phase lasts. We design our PCFG to capture changes between different flight phases, but it does not model the duration of each phase. To recover flight phase duration information, we develop flight phase duration estimates and provide these to our prediction method.

Our main contributions in this chapter are: introduction of a stochastic grammar, specifically the Probabilistic Context-Free Grammar (PCFG), to model hypersonic glide vehicle trajectories; demonstration that our method achieves accurate HGV flight phase prediction with a limited amount of training data; and use of mission types and phase duration estimates to improve flight phase prediction.

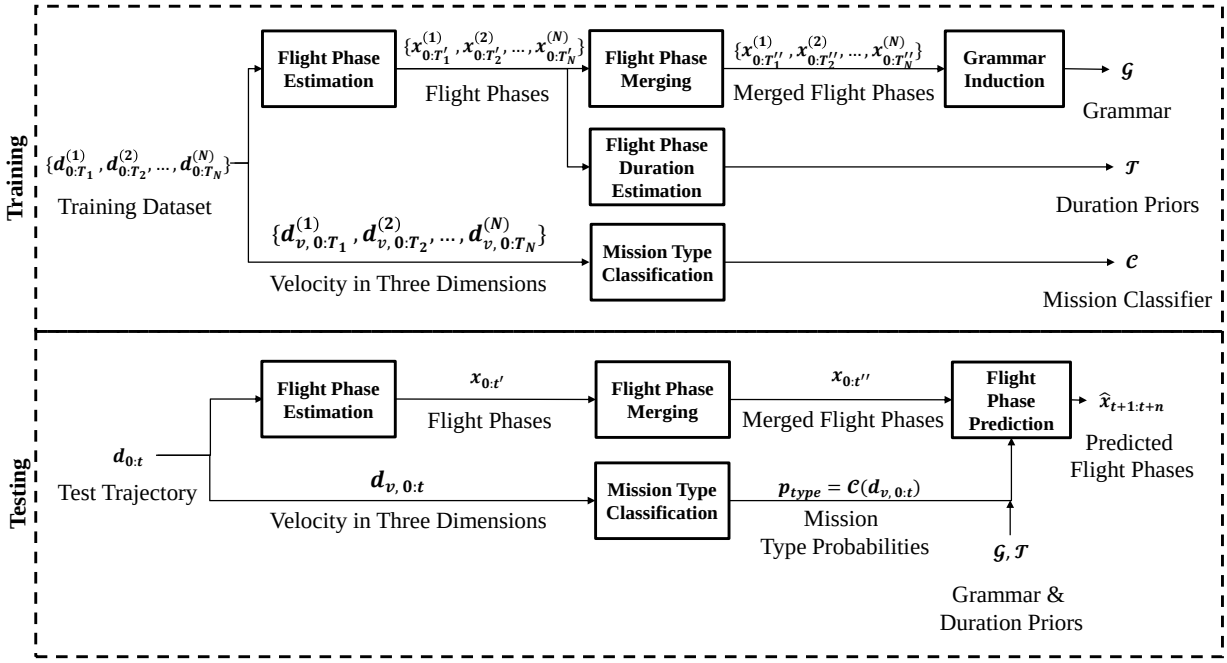


Figure 7.4. Block Diagram of Proposed Approach. Training produces: a stochastic grammar $\mathcal{G}$; a set of phase duration priors $\mathcal{T}$; and a mission classifier $\mathcal{C}$, given a collection of full-length training trajectories, denoted as $\left\{d^{(1)}_{0:T_1}, d^{(2)}_{0:T_2}, \ldots, d^{(N)}_{0:T_N}\right\}$. Note that only 3-D velocity of the training trajectories $\left\{d^{(1)}_{v,0:T_1}, d^{(2)}_{v,0:T_2}, \ldots, d^{(N)}_{v,0:T_N}\right\}$ is used to train $\mathcal{C}$, while $\mathcal{G}$ and $\mathcal{T}$ use all of the trajectory data available. During testing, a partially observed trajectory $d_{0:t}$ is used to predict flight phases (both transitions and durations) $\hat{x}$ for n future seconds of flight, given the stochastic grammar $\mathcal{G}$; mission type probabilities p_{type} from the mission classifier $\mathcal{C}$; and phase duration priors $\mathcal{T}$. We denote the resulting sequence of flight phase predictions as $\hat{x}_{t+1:t+n}$.

7.3 Proposed Approach

7.3.1 Overview of Our Approach

Figure 7.4 shows an overview of our proposed approach for flight phase prediction. During training, we use a set of HGV trajectories that trace flight paths from the point where a vehicle inserts itself into the atmosphere until the point where it reaches its final destination. In the rest of this chapter, we will refer to these trajectories as full-length trajectories (as compared to trajectories used during testing where only the initial portion of flight is available

for analysis). Let $d^{(i)}_{0:T_i}$ represent a full-length trajectory that flies from insertion at time 0 to touchdown at time T_i (measured in seconds), where i indicates the particular trajectory in the dataset; N is the total number of trajectories in the dataset; and $i = \{1, 2, ..., N\}$. Because each trajectory lasts for a different period of time, we denote the end time of each trajectory as T_i. The entire training set can be represented as $\left\{d^{(1)}_{0:T_1}, d^{(2)}_{0:T_2}, \ldots, d^{(N)}_{0:T_N}\right\}$. These full-length trajectories are used to train the components of our approach, which involves Flight Phase Estimation, Flight Phase Merging, Grammar Induction, Mission Type Classification, and Flight Phase Duration Estimation. The Flight Phase Estimation process converts trajectory data into flight phases. In this work, we define five categories of flight phases based on magnitude of rate of change of total energy of a HGV. We present the flight phase definitions in Section 7.3.3. Once we convert a trajectory into this new representation (in terms of flight phases), we denote it as $x^{(i)}_{0:T_i}$, where i still indicates the trajectory number. The trajectory has a new length, indicated by T_i, resulting from a subsampling operation done in Flight Phase Estimation. Section 7.3.3 also reviews the details of the subsampling process. Next, Flight Phase Merging combines consecutive flight phases belonging to the same flight phase category. After Flight Phase Merging, a trajectory is denoted as $x_{0:T}$, where $T \leq T$. $x_{0:T}$ contains the same flight phase categories as $x_{0:T}$ but has fewer elements. By merging the flight phases, we ensure the Grammar Induction module (described in Section 7.3.5) derives a grammar $\mathcal{G}$ that models flight phase transitions (*i.e.*, which flight phases are exhibited in what order) and not how long each of the phases occurs. Grammar Induction produces the grammar $\mathcal{G}$. To recover the temporal duration of the flight phases, Flight Phase Duration Estimation is used. Flight Phase Duration Estimation results in a set of phase durations, known as $\mathcal{T}$, that we use as prior information. Decoupling the predictions of flight phase transitions and flight phase durations reduces the complexity of the grammar $\mathcal{G}$, which improves the performance of Flight Phase Prediction. Mission Type Classification identifies whether a HGV pursues a short-range or long-range flight path based on a trajectory's velocity in three dimensions, denoted as $d^{(i)}_{v,0:T_i}$. Incorporating information about the type of mission a HGV pursues into our prediction method can improve Flight Phase Prediction because different types of missions exhibit distinct patterns in changes in energy. Mission

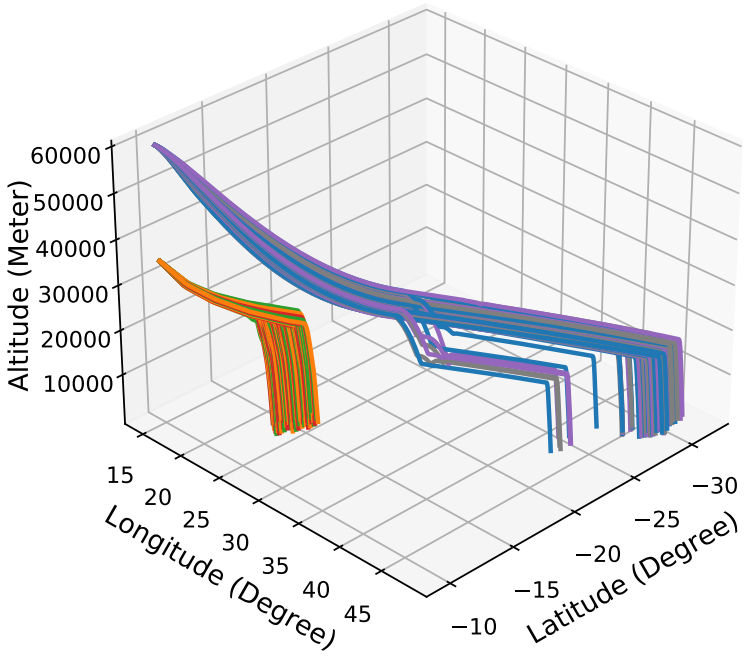
Type Classification produces a mission classifier $\mathcal{C}$. The grammar $\mathcal{G}$, flight phase duration priors $\mathcal{T}$, and mission classifier $\mathcal{C}$ are used for Flight Phase Prediction.

During testing, the proposed method analyzes a partially observed trajectory to predict flight phases for n future seconds. Let $d_{0:t}$ represent a partially observed trajectory from time 0 to t. We represent the sequence of predicted phases as $\hat{x}_{t+1:t+n}$. To obtain these predicted phases, we use Flight Phase Estimation and the testing trajectory $d_{0:t}$, resulting in a sequence of observed phases referred to as $x_{0:t}$. Flight Phase Merging produces a sequence of merged flight phases: $x_{0:t}$. We also provide 3-D velocity of a testing trajectory $d_{v,0:t}$ to the Mission Type Classification module, which produces a set of two probabilities (that we denote as p_{type}) indicating the likelihoods that trajectory $d_{0:t}$ pursues a short-range and a long-range mission. Finally, $x_{0:t}$, p_{type}, $\mathcal{G}$, and $\mathcal{T}$ are used in Flight Phase Prediction to predict future flight phases $\hat{x}_{t+1:t+n}$.

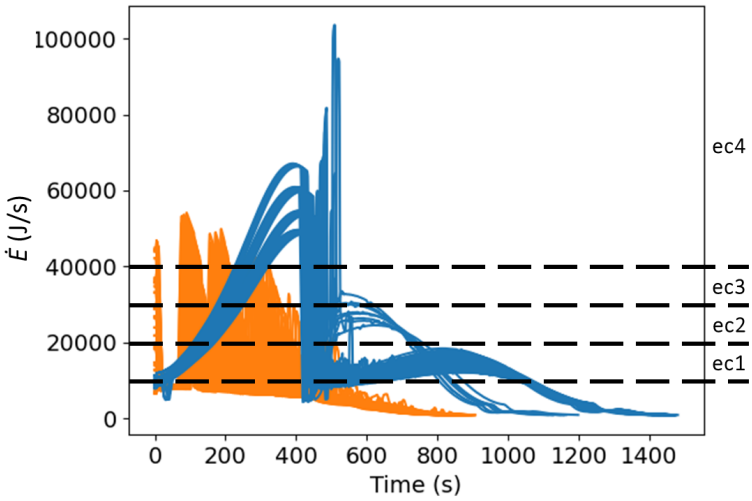
7.3.2 Dataset and Motivation for Using Magnitude of Rate of Change of Total Energy

Our dataset consists of 150 simulated HGV trajectories. Seventy-five trajectories are short-range HGV missions, and the other seventy-five trajectories are long-range HGV missions. Short-range and long-range missions are differentiated based on the distance a HGV flies in terms of downrange (*i.e.*, distance from initial launch point). We focus our analysis on the portions of the trajectories that occur after a HGV's insertion into the atmosphere. Figure 7.5a illustrates the flight paths of the trajectories used in our experiments. The 3-D coordinates (longitude, latitude, and altitude values, in this case) show that all trajectories commence from the same longitude-latitude location after insertion into the atmosphere and that short-range missions start at a lower altitude than long-range missions. Depending on the length of the mission and maneuvers performed, a vehicle may fly for longer or shorter periods of time.

The trajectories are simulated using a waypoint-following navigation scheme. Given a set of waypoints, the navigation logic selects a maneuver that best propagates a HGV towards the next waypoint. Our trajectory generation tool selects the best maneuver from a discrete set of potential maneuvers, where the potential maneuvers are defined by changes to two



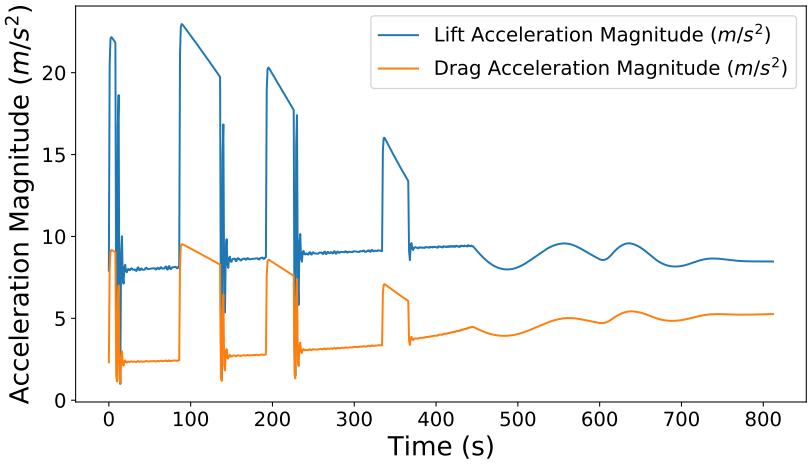
(a) 3-D flight paths of trajectories in our dataset. Trajectories in warm colors (*e.g.*, orange, yellow, red) are short-range missions, while trajectories in cold colors (*e.g.*, blue, purple) are long-range missions.



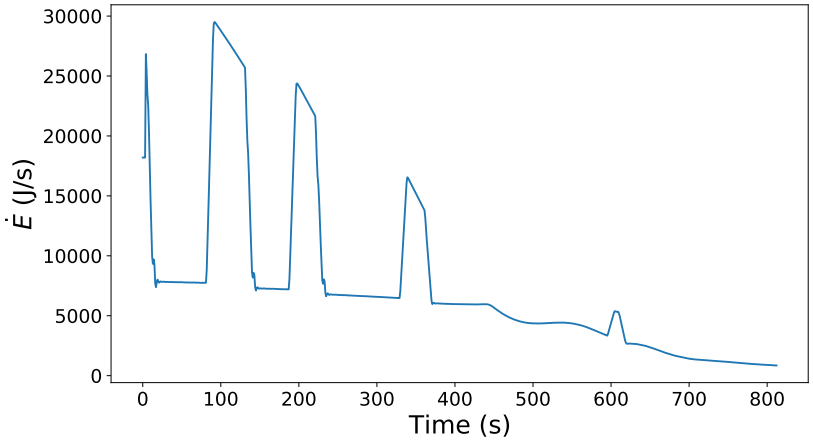
(b) Magnitude of rate of change of total energy (denoted as $\dot{E}$) of all trajectories. Short-range trajectories appear in orange, and long-range trajectories appear in blue. Dashed, horizontal lines (in black) indicate different flight phases.

Figure 7.5. HGV Dataset. Our simulated dataset contains an equal number of short-range and long-range trajectories.

control inputs: angle of attack and bank angle. The tool utilizes a ballistic re-entry model with the aerodynamic profile of a HGV. The ballistic re-entry model is a set of ordinary differential equations that define a dynamical model with gravity, lift, and drag for a vehicle moving through Earth's atmosphere [201]. The hypersonic glide vehicle has the aerodynamic profile of a Common Aero Vehicle-H (CAV-H), a specific type of HGV [202], [203]. We obtain the aerodynamic profile for our HGV from [204], [205]. The profile consists of lift and drag coefficients as a function of angle of attack and Mach number.



(a) Flight maneuvers (lift and drag accelerations) for a HGV trajectory.



(b) Magnitude of rate of change of total energy (denoted as $\dot{E}$) of a HGV trajectory.

Figure 7.6. Comparison of HGV Features. Magnitude of rate of change of total energy exhibits a similar pattern (*e.g.*, same raising and dropping) as lift and drag accelerations.

Lift and drag accelerations imply how a vehicle might maneuver in the future. Directly estimating lift and drag accelerations in real-world situations can result in noisy and unreliable estimates, though, which would inhibit our predictive methods. Thus, we use kinematic features (specifically 3-D velocity and altitude) that can be estimated with higher reliability to determine an alternative aerodynamic parameter that implies how a vehicle might maneuver: magnitude of rate of change of total energy. Figure 7.6a shows lift and drag accelerations from a single trajectory in our dataset, and Figure 7.6b shows magnitude of rate of change of total energy for the same trajectory. This side-by-side comparison reveals that both exhibit analogous behavior, so we use magnitude of rate of change of total energy to imply information about lift and drag. The magnitude of rate of change of total energy also indicates which maneuvers a vehicle could perform, based on physics. Vehicles rely upon limited energy sources based on their velocities and initial diving altitudes. As vehicles fly longer and execute more maneuvers, they use more and more of their energy supply. At some point, they can no longer perform further actions because they lack the energy to do so. By using magnitude of rate of change of total energy, we provide physics-based information to our method to predict future flight phases. We use 3-D velocity and altitude to find the magnitude of rate of change of total energy. We simulate these kinematic features (*i.e.*, 3-D velocity and altitude) at a 1 Hz rate in Earth-Centered Inertial (ECI) coordinates for all trajectories in our dataset. Then, the flight phases of the trajectories are estimated (as shown in Section C) using magnitude of rate of change of total energy.

Figure 7.5b shows magnitude of rate of change of total energy of all trajectories in our dataset. We designed this multi-mission dataset using a realistic scenario in predicting future flight phases. The short-range trajectories and long-range trajectories have distinct energy change rate characteristics. Predicting future flight phases is more difficult for short-range missions since their flight phase transitions are more complex than long-range missions, especially earlier in flight (*e.g.*, 0 - 400 seconds). To overcome these challenges, we leverage mission-type information from the mission type classifier $\mathcal{C}$ in our flight phase predictor to improve the performance of flight phase prediction.

7.3.3 Flight Phase Estimation

We use Flight Phase Estimation to convert trajectory data (*i.e.*, kinematic features) into sequences of flight phases. Flight Phase Estimation labels the trajectories in our dataset, determining which phases they exhibit throughout flight. We define five different categories of flight phases (known as $ec0$, $ec1$, $ec2$, $ec3$, and $ec4$) based on magnitude of rate of change of total energy $\dot{E}$. Recall that x_t is a flight phase at time t of a trajectory. Thus, $\dot{E}_t$ represents magnitude of rate of change of total energy of a trajectory at time t. Let the set Ω denote the flight phases, where $x_t \in \Omega$. We define Ω as:

$$\Omega = \begin{cases} ec0, & \dot{E}_t \leq 10^4, \\ ec1, & 10^4 < \dot{E}_t \leq 2 \times 10^4, \\ ec2, & 2 \times 10^4 < \dot{E}_t \leq 3 \times 10^4, \\ ec3, & 3 \times 10^4 < \dot{E}_t \leq 4 \times 10^4, \\ ec4, & \dot{E}_t > 4 \times 10^4. \end{cases} \tag{7.1}$$

From this definition, $ec0$ represents the smallest magnitude of rate of change of total energy, and $ec4$ represents the largest magnitude of rate of change of total energy.

We define the thresholds in Equation 7.1 to ensure each flight phase category is equally represented in our dataset. The thresholds and flight phase distributions are shown in Figure 7.5b where the blue lines are the thresholds. The proposed method does not require an equal distribution of flight phases to perform well, but the methods we use for comparison in our experiments do. We compare our PCFG to two popular deep learning-based methods: LSTM [81] and Seq2Seq model [127]. These data-driven methods perform best on datasets with balanced classes. In unbalanced cases, the methods tend to ignore the classes that occur infrequently in training. To fairly compare our method against these models, we choose thresholding values that result in a balanced dataset in terms of flight phases.

Figure 7.7 shows an overview of the Flight Phase Estimation process. We begin with a sequence of kinematic features that describes a trajectory having T seconds of flight. Let $d_{0:T}$ represent this trajectory with kinematic features:

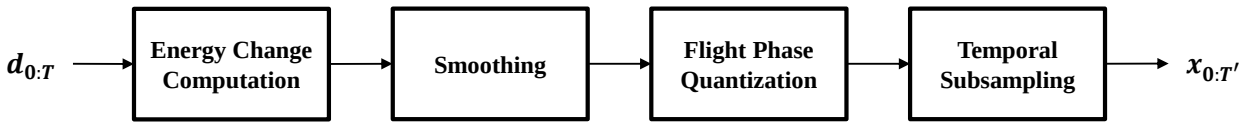


Figure 7.7. Block Diagram of Flight Phase Estimation. Flight Phase Estimation uses a trajectory represented as a sequence of kinematic features (*i.e.*, 3-D velocity and acceleration) – denoted as $d_{0:T}$, where $d_{0:T}$ ($0{:}T := [0, 1, 2, ..., T]$) – to find a new representation of the trajectory as a sequence of flight phases – denoted as $x_{0:T}$, where $0{:}T := [0, 10, 20, ..., T]$.

- $h_{0:T}$: altitude;
- $v_{x,0:T}$: velocity in the ECI x direction;
- $v_{y,0:T}$: velocity in the ECI y direction; and
- $v_{z,0:T}$: velocity in the ECI z direction,

where $0{:}T := [0, 1, 2, ..., T]$. Flight Phase Estimation uses these kinematic features to determine magnitude of rate of change of total energy over the entire trajectory, resulting in sequence $\dot{E}_{0:T}$. First, we find potential energy and kinetic energy for every second of flight, resulting in $E_{p,0:T}$ and $E_{k,0:T}$, respectively. For one second of flight, potential energy at time t (where $0 \leq t \leq T$) is calculated as:

$$E_{p,t} = m g_t h_t, \tag{7.2}$$

and kinetic energy at time t is calculated as:

$$E_{k,t} = \frac{1}{2} m |v_t|^2, \tag{7.3}$$

where g_t is acceleration due to gravity at time t and $|v_t|$ is velocity magnitude at time t. Velocity magnitude at time t is calculated as:

$$|v_t| = \sqrt{v_{x,t}^2 + v_{y,t}^2 + v_{z,t}^2}. \tag{7.4}$$

We obtain g_t from a WGS84 spherical Earth gravitational acceleration model [206]. For mass m, we use $m = 1$. Thus, we calculate specific energy (*i.e.*, total of kinetic and potential energy per unit mass) of the HGV. We do not need the exact mass of the vehicle for our approach since we define flight phases based on relative differences in magnitude of rate of change of total energy, so specific energy is sufficient for our purposes. In the remainder of the chapter, we will refer to the specific energy of the HGV as total energy. Total energy at time t is:

$$E_{total,t} = E_{p,t} + E_{k,t}. \tag{7.5}$$

We define the magnitude of rate of change of total energy at time t as:

$$\dot{E}_t = \left| \frac{(E_{total,t} - E_{total,t-1})}{\Delta t} \right|, \tag{7.6}$$

where $\Delta t = 1$ since our data has 1-second timesteps. We calculate $\dot{E}_t$ for the entire sequence $d_{0:T}$ to form a new representation of the trajectory as a sequence of magnitude of rate of change of total energy $\dot{E}_{0:T}$.

$\dot{E}_{0:T}$ can be noisy because a derivative operation (or in our case, a finite difference approximation) enhances noise. We smooth $\dot{E}_{0:T}$ with a 9-second moving average filter to obtain $\dot{E}_{smoothed,0:T}$. The smoothing operation aids in quantization of the flight phases. Finally, we quantize $\dot{E}_{smoothed,0:T}$ to obtain flight phases using the set Ω in Equation 7.1. We quantize the trajectories in this way because stochastic grammars use discrete labels. $x_{0:T}$ denotes a trajectory in this new representation (*i.e.*, in terms of flight phases).

The original temporal resolution of $x_{0:T}$ is one second. As shown in Figure 7.5b, many long-range trajectories extend more than $1,400$ seconds in length. These long sequences can result in extensive computation times if we process every second of the data. Therefore, we downsample each trajectory by a factor of ten to shorten the trajectory lengths and reduce computation times. We use a non-overlapping window to subsample the trajectories. First, a trajectory $x_{0:T}$ is divided into 10-second non-overlapping segments. Then, the most common flight phase (*i.e.*, the mode) in each 10-second segment serves as new flight phase label for that portion of the trajectory. Let $x_{0:T}$ represent this new trajectory, where $0{:}T$

$:= [0, 10, 20, ..., T]$. In practice, any downsampling rate can be used, depending on the application and computation budget.

7.3.4 Flight Phase Merging

A trained stochastic grammar $\mathcal{G}$ contains production rules used to predict future flight phases. We force the production rules to focus on flight phase transitions rather than flight phase durations (we will recover the flight phase durations later) by utilizing Flight Phase Merging. In this process, consecutive flight phases with the same phase label are combined into one element. For example, consider a flight phase sequence $x_{0:T}$ such as $ec0$ $ec0$ $ec0$ $ec1$. Flight Phase Merging collapses the repeated flight phase (*i.e.*, $ec0$) into a single element, resulting in a new flight phase sequence $x_{0:T}$ of two elements: $ec0$ $ec1$. Flight Phase Merging preserves information about flight phases exhibited in trajectories while reducing the number of elements in the trajectories (T denotes the new length of the flight phase sequence after merging). Now, the grammar does not need to learn rules that describe how often a flight phase repeats (*e.g.*, $ec0$ $ec0$ $ec0$ versus $ec0$ $ec0$), which reduces the complexity of the resulting grammar. Instead, the grammar $\mathcal{G}$ learns phase transitions (*i.e.*, changes between two different types of flight phases).

7.3.5 Grammar Induction

After converting the trajectories to flight phases, a stochastic grammar $\mathcal{G}$ is derived that describes them. A brief review of the stochastic grammar known as the Probabilistic Context-Free Grammar (PCFG) [199] is presented here. Then, the Grammar Induction method used in this chapter, Automatic Distillation of Structure (ADIOS) [207] is described. ADIOS learns a grammar $\mathcal{G}$ that describes the general patterns (*i.e.*, flight phase transitions) of HGV trajectories.

Formally, PCFG is defined by a quintuple $\mathcal{G} = (\Omega, V, T, R, P)$, where

- Ω is a finite set of terminal symbols;
- V is a finite set of non-terminal symbols;

- $T \in V$ is the start symbol;
- R is a finite set of production rules;
- P is a finite set of probabilities.

Each of these components are described in detail below. It is helpful to refer to an example parse graph to better understand them, so Figure 7.3 is included here again.

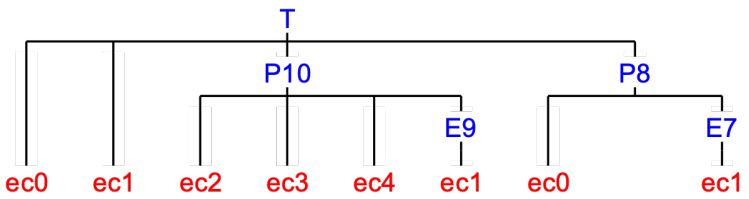


Figure 7.8. A HGV Trajectory as a Parse Graph. The parse graph represents a trajectory as a "sentence", where nodes and edges of the parse graph show the sentence structure of the trajectory. Flight phase labels (*i.e.*, "words") are shown in red; groups of flight phase labels (*i.e.*, higher-level "phrases" in a sentence) are shown in blue; and flight phase transition patterns (*i.e.*, production rules in a grammar) are shown as edges connecting the nodes in the parse graph.

In Figure 7.8, the terminal symbols are shown in red and are located in the final layer of the parse graph. Terminal symbols are the smallest units of a sequence under analysis. A terminal symbol cannot be decomposed into a smaller unit. In a language modeling task, terminal symbols could be words. In our case, terminal symbols are flight phases based on magnitude of rate of change of total energy. Thus, the set of terminal symbols is $\Omega = \{ec0, ec1, ec2, ec3, ec4\}$.

Non-terminal symbols can be decomposed further. Every non-terminal symbol in V can be expanded into a new sequence of terminal and non-terminal symbols. Non-terminal symbols are located in layers above the final layer of a parse graph. Figure 7.8 shows them in blue. In a language modeling task, non-terminal symbols could describe phrases (*i.e.*, groups of words) in sentences, such as noun phrases and verb phrases. In our case, each non-terminal symbol describes a unique relationship between components of a trajectory that corresponds to flight phase transition patterns. T is a special type of non-terminal symbol

because it serves as the root node for all parse graphs, representing a trajectory before its been decomposed.

Production rules in R define how non-terminal symbols and terminal symbols relate. More specifically, production rules indicate how a non-terminal symbol expands into a sequence of non-terminal and/or terminal symbols. Here are a few examples of production rules based on the trajectory shown in Figure 7.8:

- $T \rightarrow ec0 \quad ec1 \quad P10 \quad P8$;
- $P10 \rightarrow ec2 \quad ec3 \quad ec4 \quad E9$;
- $E9 \rightarrow ec1$;
- $P8 \rightarrow ec0 \quad E7$;
- $E7 \rightarrow ec1$;

where $T, P10, P8, E9, E7 \in V$ and $ec0, ec1, ec2, ec3, ec4 \in \Omega$. The symbol found on the left side of $\rightarrow$ is the non-terminal symbol that can be expanded. On the right side of $\rightarrow$ is the expanded sequence of non-terminal and/or terminal symbols. For example, $P8 \rightarrow ec0 \quad E7$ means a non-terminal symbol $P8$ can be decomposed into smaller constituents (*i.e.*, $ec0$ and $E7$). Note that the order of the new symbols sequence is meaningful. In other words, $P8 \rightarrow ec0 \quad E7$ and $P8 \rightarrow E7 \quad ec0$ are two different production rules. Multiple different expansions of the same non-terminal symbol are permitted as well. For example, $P8 \rightarrow ec0 \quad E7 \mid ec0 \quad E9$ means that $P8$ can be expanded to $ec0 \quad E7$ or to $ec0 \quad E9$. In this chapter, we use upper case English letters to represent non-terminal symbols V.

The final component of $\mathcal{G}$ is a set of probabilities P. The probabilities in P correspond to each production rule in $\mathcal{G}$. Similar to [191], we determine these probabilities according to:

$$P(A \rightarrow \bigstar) = \frac{\#(A \rightarrow \bigstar)}{\#(A \rightarrow *)}, \tag{7.7}$$

where $A \in V$ is a non-terminal symbol; $\bigstar$ is an expansion of the non-terminal symbol A; $A \rightarrow \bigstar$ is the production rule for which the probability is being calculated; $\#()$ represents the total number of occurrences in the training set of the production rule inside the parentheses;

and $A \rightarrow *$ represents all production rules that start with the same non-terminal symbol as the production rule for which the probability is being calculated. In other words, this equation calculates the probability of a rule occurring as the total number of observations of that rule in the training set divided by the total number of rules beginning with the same start symbol in the training set. For example, to calculate the probability associated with the rule $P8 \rightarrow ec0 \quad E7$, Equation 7.7 would become:

$$P(P8 \rightarrow ec0 \quad E7) = \frac{\#(P8 \rightarrow ec0 \quad E7)}{\#(P8 \rightarrow *)}. \tag{7.8}$$

Given the flight phases obtained from the training set, we use Grammar Induction to learn a set of production rules that describe the general patterns of flight phase transitions. As in [191], we use the Automatic Distillation of Structure model (ADIOS) [207] to obtain these rules. ADIOS produces two types of rules:

- P-Rule: a rule that captures a major, statistically significant pattern. It captures a subsequence of flight phases that appears frequently in the training set. One example of a P-rule is $P1 \rightarrow ec0 \quad ec1 \quad ec2$. Note that we use $P*$, where $*$ represents different indices, to distinguish different P-rules.

- E-Rule: a rule that defines more generalizable rules. It allows rules with multiple expansions (*i.e.*, equivalent options). An example of an E-rule is $E1 \rightarrow ec0 \quad | \quad ec1$, which indicates that the non-terminal symbol $E1$ can be replaced by either $ec0$ or $ec1$. Again, we use $E*$ (where $*$ represents different indices) to distinguish different E-rules.

Nested rules, such as $P1 \rightarrow ec0 \quad E1 \quad ec2$ and $E1 \rightarrow ec1 \quad | \quad P2$, are permitted in ADIOS. ADIOS iteratively determines the set of all P-rules and E-rules to generalize the grammar, iterating until the rule set no longer changes. Please refer to the original paper [207] for the details of ADIOS.

7.3.6 Mission Type Classification

As shown in Figure 7.5b, short-range missions and long-range missions have distinct energy characteristics. Energy of a HGV will rise and fall in a manner unique to each

mission. Therefore, including mission information in our prediction method can help the method choose production rules from the grammar that are more suitable to the mission being flown. To incorporate the mission type information in our approach, we use a 1-layer LSTM [81] for Mission Type Classification. Given 3-D velocity of a partially observed trajectory $d_{v,0:t}$, the LSTM predicts two probabilities: $p_{type} = \mathcal{C}(d_{v,0:t}) \in \mathbb{R}^2$, where $\mathcal{C}$ is the LSTM model with a softmax function as the last layer and p_{type} is the predicted probability vector for short-range and long-range missions. Note that 3-D velocity $d_{v,0:t}$ is not the same as velocity magnitude $|v_{0:t}|$. 3-D velocity $d_{v,0:t}$ is a 2-D matrix with dimensions 3 x t, where the three separate rows of the matrix contain $v_{x,0:t}$, $v_{y,0:t}$, and $v_{z,0:t}$, respectively. Although our dataset is challenging for flight phase prediction, it exhibits distinct trajectory paths for each of the two mission types. Thus, Mission Type Classification is relatively simple. Even with a simple 1-layer LSTM, we achieve 100% accuracy by "observing" only 100 seconds of flight from the initial portions of trajectories in the training dataset (*i.e.*, $t = 100$). When dealing with complex trajectories of multiple types of vehicles, a more effective vehicle classification method can be used, such as that described in [208].

With the mission type probability p_{type}, we can modify the probabilities of the PCFG production rules in P – calculated with Equation 7.7 – to incorporate the mission type information as follows:

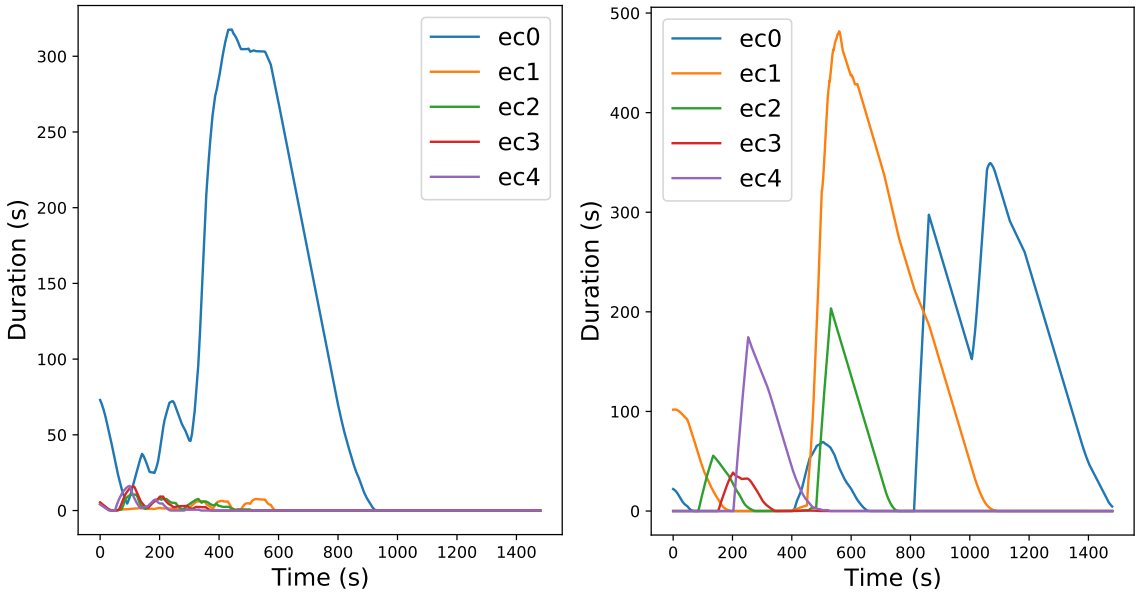
$$\hat{P}(r) = P(r) \cdot (p_{type}^T \cdot w_{type}(r)), \tag{7.9}$$

where $w_{type}(r) \in \mathbb{R}^2$ are the weights of a rule r belonging to each mission type. $w_{type}(r)$ is defined as:

$$w_{type}(r) = \begin{bmatrix} \frac{\#(r|D_{short})}{\#(r|D_{all})} \\ \\ \frac{\#(r|D_{long})}{\#(r|D_{all})} \end{bmatrix}, \tag{7.10}$$

where $\#(r|D_{short})$ counts the number of occurrences of rule r in the short-range trajectories in training data D_{short}; $\#(r|D_{long})$ counts the number of occurrences of rule r in the long-range trajectories in training data D_{long}; and $\#(r|D_{all})$ counts the total number of occurrences of rule r in the entire training dataset D_{all}. Therefore, the product $p_{type}^T \cdot w_{type}(r)$ has a large value only if the weight and type probability for a certain type of mission are large, simultaneously.

7.3.7 Flight Phase Duration Estimation



(a) Duration priors for short-range missions. (b) Duration priors for long-range missions.

Figure 7.9. Duration Priors. Given the mission type, predicted phase, and current time, the phase duration priors output the duration of the predicted phase.

We do not design the PCFG to estimate the duration of each phase. To recover the duration information, a set of flight phase duration priors based on the flight phases in the training data is found. As shown in Figure 7.9, we use the flight phase obtained from the Flight Phase Estimation module to determine the duration of each flight phase and each mission type based on the trajectories in the training set. A flight phase's duration is found as the average of the flight phase duration for every time t. Figure 7.9a shows that the majority of the short-range trajectories end after 900 seconds and that most of the phases are $ec0$, which is consistent with the magnitude of rate of change of total energy shown in Figure 7.5b. As shown in Figure 7.9b, the long-range trajectories have a more diverse flight phase distribution at different times. Figure 7.5b shows that these trajectories usually have slow transitions, meaning that the phase durations are typically longer than the short-range trajectories. Given a predicted flight phase l (the details of flight phase prediction process will be provided in the next section), the predicted mission type classification $c = \arg\max(p_{type})$,

and the current time t, we obtain the duration of the predicted phase as $\mathcal{T}_c(t, l)$, where $\mathcal{T}_c$ is the flight phase duration prior of mission type c as shown in Figure 7.9. We call these values "priors" because flight phase duration information is collected solely based on the training data. During testing, the flight phase duration information can be obtained from these priors without knowledge of the testing data.

7.3.8 Phase Prediction

To predict future phases of flight and the phase durations, the proposed flight phase predictor uses:

- flight phases after merging, $x_{0:t}$;
- the stochastic grammar (*i.e.*, PCFG) $\mathcal{G}$;
- the set of flight phase duration priors $\mathcal{T}$; and
- the mission type probability p_{type}.

Figure 7.4 shows an overview of this process. Similar to [192], [193], we use Generalized Earley Parser (GEP) for predictions. In [191], the original Earley parser [209] is used to predict human activity. The Earley parser was designed for parsing sentences using a language grammar. The Earley parser reads terminal symbols (*i.e.*, words in a sentence or, in our case, flight phases) sequentially and returns all possible parse graphs given the grammar $\mathcal{G}$. Due to grammar ambiguity, a symbol sequence (*e.g.*, flight phase sequence) can have multiple parse graphs. In [191], it was proposed to use the pending states (*i.e.*, the rules that have not been fully used after reading the observed flight phases) for predictions. However, as shown in [192], [193], the original Earley parser is sensitive to the input sequence. Changing one flight phase in the input sequence can yield completely different parse graphs, which will cause different predictions. If the prediction method needs to operate in a noisy environment, a parser more resistant to noise is required.

To solve this input sensitivity issue, the Generalized Earley Parser (GEP) [192], [193] uses a matrix of flight phase probabilities as an input instead of a flight phase sequence $x_{0:t}$.

GEP will find the flight phase sequence from the probability matrix that best describes the trajectory given the PCFG grammar rules. For example, in our 5-category flight phase case, the probability matrix for a N-phase sequence has the shape of $5 \times N$, where each column is the probabilities of the flight phase occurring for each class. In this chapter, we define the conversion from flight phase sequence to probability matrix as follows:

$$p_i(l) = \begin{cases} \frac{1}{1+4\epsilon} & l = i \\ \\ \frac{\epsilon}{1+4\epsilon} & l \neq i \end{cases}, \forall i \in \Omega,$$

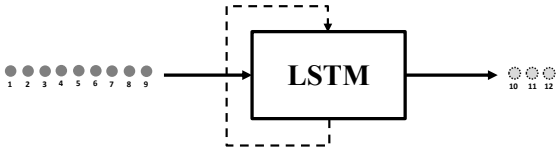
where l is a given query flight phase. As suggested by [193], we set $\epsilon = 10^{-10}$. The use of this probability matrix enables GEP to handle noise in flight phase sequences. Since we define all values in the probability matrix as positive values, GEP will find the best flight phase sequence based on the PCFG grammar from all possible phase sequences, as shown in [192], [193]. Therefore, if a flight phase is incorrectly changed due to noise (*e.g.*, from $ec0 \quad ec1 \quad ec2$ to $ec0 \quad ec4 \quad ec2$), GEP will still find the correct flight phase sequence ($ec0 \quad ec1 \quad ec2$) based on the PCFG rules and their corresponding probabilities. Furthermore, the use of GEP for flight phase prediction enables a more explainable prediction approach compared to deep learning-based methods, such as LSTM [81] or Seq2Seq [127]. Given a predicted flight phase, GEP is able to trace back to the original production rule that predicted this particular flight phase. As shown in Figure 7.8, the production rules construct the parse graph of a trajectory, which provides an effective representation/visualization to help understand the structure of the entire trajectory. Production rules can be explored in this manner to understand a prediction for HGV flight in a more explainable way. Please refer [192], [193] for the details of GEP.

7.4 Experimental Setup

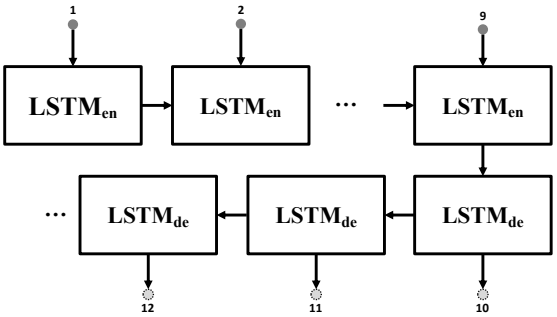
7.4.1 Dataset Partitioning and Discussion

We evaluate performance assuming a scenario of limited training data. We have 150 trajectories in total, including 75 short-range trajectories and 75 long-range trajectories. We

divide the full dataset into training and testing sets according to a 2:8 ratio. We purposefully limit the amount of training data to evaluate performance of all methods with minimal examples from which to learn. More specifically, we use 30 trajectories (15 short-range and 15 long-range) as training data, while the remaining 120 trajectories (60 short-range and 60 long-range) serve as the testing data. For now, we proceed with this data split. Later, we show results with different ratios of training and testing data to analyze performance related to the amount of training data available.



(a) Block diagram of the LSTM model.



(b) Block diagram of the Seq2Seq model.

Figure 7.10. Comparison Methods. We compare the performance of our proposed stochastic grammar to these two LSTM-based methods.

7.4.2 Comparison Methods

We consider two alternative methods, shown in Figure 7.10, to verify our approach. Specifically, we utilize a LSTM [81] and a Seq2Seq model [127]. We use these methods for comparison since they often succeed in prediction tasks with sequential data [210], [211].

We design the baseline LSTM [81] as one recurrent layer with 32 features in the hidden state. The LSTM iterates over the flight phases in the input sequence (*i.e.*, sequence of observed flight phases) and predicts future flight phases until it reaches the desired prediction duration. We use a shared, fully connected layer to map each output state of the recurrent layer to the flight phase space. The output state at each predicted timestep has shape $B \times D$, where B is batch size and D is the size of the hidden state. The flight phase space has shape $B \times 5$ since we use 5 flight phase classes defined in Equation 7.1. A softmax function

determines the predicted flight phase at a specific timestep from the 5-class predicted flight phase space.

As shown in Figure 7.10b, the Seq2Seq [127] consists of two LSTMs: an encoder and a decoder. The encoder first iterates over the flight phases in the input sequence. It produces a hidden state of 32 features. Then, the hidden state serves as an input to the decoder, which predicts future flight phases with a shared fully connected layer to map the output state to the flight phase space. Again, the Seq2Seq model continues to make predictions until the desired prediction duration is reached.

For a fair comparison, we evaluate the LSTM and Seq2Seq methods with and without mission type probability. More specifically, for the experiments without mission type probability, we only provide the flight phase sequence (without Flight Phase Merging to keep the flight phase duration information) to the LSTM and Seq2Seq models. Therefore, the input is $B \times L \times 1$, where B is batch size and L is observed sequence length. For the experiments with mission type probability, we concatenate the input flight phase sequence with the mission type probability. Note that we only include the short-range probability, since the long-range probability is its complement. Therefore, the input in this case is $B \times L \times 2$.

To train the LSTM and Seq2Seq models, we use a batch size of eight. Additionally, we use the Adam optimizer [78] with an initial learning rate of 10^{-4}. The learning rate decays linearly over the training period. Both models are trained with cross entropy loss [212] for 100 epochs.

7.4.3 Experimental Results

In this work, we use flight phase prediction accuracy to compare the performance of the PCFG, LSTM, and Seq2Seq models. Table 7.1 shows a summary of results of all methods we evaluated when they are trained on only 20% of the full dataset. From this table, we observe that the proposed PCFG performs significantly better than both LSTM and Seq2Seq in all scenarios. First, we consider the most challenging case: 100 seconds of observed data without mission type information. Our PCFG achieves 0.6708 accuracy in this scenario, while the LSTM and Seq2Seq achieve only 0.2908 and 0.4283, respectively. The PCFG

clearly outperforms LSTM and Seq2Seq by a significant margin, which shows that PCFG can perform well in this limited training data scenario. As the length of observation increases, the performance of PCFG increases and continues to surpass the two baseline methods. For example, the PCFG obtains its best performance without mission type information (0.8567 accuracy) when it observes 300 seconds of a trajectory. However, the LSTM and Seq2Seq only manage 0.4183 and 0.6358 accuracies, respectively. The PCFG performs well in all cases even though the mission type information is not available.

Table 7.1. Flight Phase Prediction Accuracy. Results with the proposed stochastic grammar PCFG, LSTM, and Seq2Seq for different lengths of observed flight phases, with and without mission type probability.

HGV Flight Phase Prediction Results				
Method	Train-Test Ratio	Observation Time (s)	Accuracy of Predicting 100s Without Type Probability	Accuracy of Predicting 100s With Type Probability
PCFG	2:8	100	**0.6708**	**0.6758**
	2:8	200	**0.5983**	**0.6308**
	2:8	300	**0.8567**	**0.8683**
LSTM	2:8	100	0.2908	0.3392
	2:8	200	0.3717	0.2875
	2:8	300	0.4183	0.4908
Seq2Seq	2:8	100	0.4283	0.2717
	2:8	200	0.3758	0.3767
	2:8	300	0.6358	0.8592

When we extend our analysis further to incorporate mission type information (but still only train on 20% of data), the PCFG also outperforms LSTM and Seq2Seq in all cases (*i.e.*, observing 100, 200, and 300 seconds). In most cases, the performance increases for PCFG, LSTM, and Seq2Seq when mission type information is available. This indicates that the mission type information can be used to improve flight phase prediction.

Overall, the improvement of PCFG using the mission type information is not significant, especially compared to the improvement of Seq2Seq in the 300-second observation scenario.

Since the grammar induction method, ADIOS, learns a set of generalized production rules for the entire training set, there is a large number of shared rules between short-range and long-range trajectories. Thus, the weights in Equation 7.10 will approach 0.5 for most of the rules, which indicates the rules are equally applicable to either type of mission. Therefore, the mission type probabilities are not effective for those rules. In general, when dealing with more mission types, there will be fewer shared rules across all trajectory types. In that scenario, the mission type probability will improve the performance more effectively.

Table 7.2. Flight Phase Prediction Accuracy with Different Amounts of Training Data. All results shown in this table were obtained by using the mission type probabilties.

HGV Flight Phase Prediction Results for Different Train-Test Ratios				
Method	Observation Time (s)	Accuracy of Predicting 100s with Train-Test Ratio of 2:8	Accuracy of Predicting 100s with Train-Test Ratio of 5:5	Accuracy of Predicting 100s with Train-Test Ratio of 8:2
PCFG	100	**0.6758**	**0.6697**	**0.5900**
	200	**0.6308**	**0.6316**	0.6400
	300	**0.8683**	0.8974	0.8667
LSTM	100	0.3392	0.3474	0.3300
	200	0.2875	0.3013	0.3800
	300	0.4908	0.4961	0.4967
Seq2Seq	100	0.2717	0.4500	0.4633
	200	0.3767	0.3868	**0.6600**
	300	0.8592	**0.9132**	**0.9100**

Now, we consider how the methods perform with different amounts of training data, assuming mission type information is used. As shown in Table 7.2, we evaluate performance when the training data is 20%, 50%, and 80% of all available data. As the amount of training data increases, we observe that the proposed stochastic grammar does not increase in performance. Instead, the PCFG maintains fairly comparable performance. More training data does not increase the PCFG performance because it can learn a set of rules that govern flight phases from a limited training set. In theory, as long as the training set contains

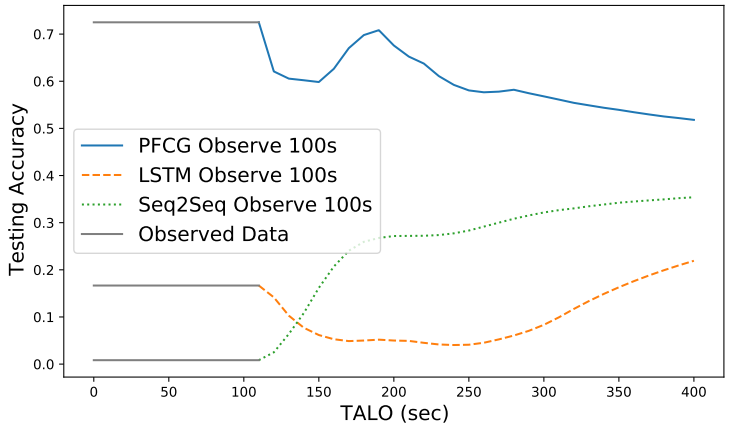
exemplary training trajectories that encompass all possible rules that could be exhibited by a HGV, the PCFG will perform well. Usually, deep learning methods require an extensive amount of training samples, though. As the size of the training set increases, these deep learning methods can perform better (*e.g.*, Seq2Seq trained on 80% of data). Therefore, we show that the proposed PCFG can achieve good performance without the requirement of a large training set, especially compared to the deep learning methods.

Next, we consider performance as a function of time, as shown in Figure 7.11. The gray, horizontal lines indicate the amount of observed data analyzed in order to inform predictions. The colored lines show accuracy of predictions for the different methods. For example, Figure 7.11a shows results when the methods observe 100 seconds of data and predict flight phases from 100s to 400s into the future. Similar to results summarized in Table 7.1, the PCFG usually achieves the best performance for various prediction times. Especially for the cases where the methods only observe 100 or 200 seconds (shown in Figures 7.11a and 7.11b, respectively), PCFG outperforms LSTM and Seq2Seq by a significant margin. Although Seq2Seq achieves comparable performance to PCFG when it observes 300 seconds of flight (shown in Figure 7.11c), the prediction task is much easier in this case than the first two scenarios, as we discuss below.

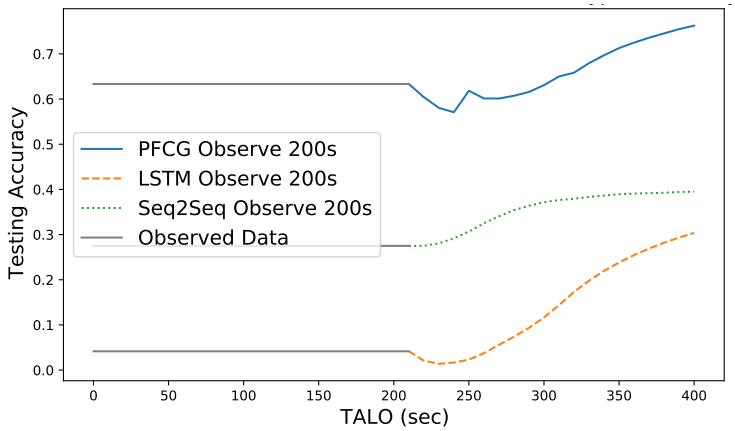
One of the best aspects of the PCFG is its ability to model flight phase transitions. Figure 7.12 shows six different trajectories in our testing set. For each trajectory, the top bar graph illustrates the ground truth (GT) flight phases (*i.e.*, the actual flight phases the trajectory exhibits during the prediction period) as a function of time. The other three bar graphs show the predicted flight phases from each method we consider (*i.e.*, PCFG, LSTM, and Seq2Seq). The gray portions of the bar graphs indicate the portions of the trajectories that serve as observation periods. The total duration in the bar graphs is 400 seconds, and the observed duration changes from 100 seconds to 300 seconds (similar to the experiments shown in Figure 7.11). These examples were obtained by training with limited data, so 20% of data was used for training. From this figure, we see that only PCFG successfully predicts flight phase transitions, while LSTM and Seq2Seq usually predict the same phase for an entire prediction period. We observe that the PCFG usually predicts flight phases in the correct order, matching the ground truth.

Let us consider a simpler example for flight phase prediction, as shown in Figure 7.12f. The ground truth indicates that methods will examine 300 seconds of observed data and should predict one phase category ($ec0$) for the next 100 seconds. The PCFG and Seq2Seq correctly predict $ec0$ in this example, but the LSTM fails. If we look at the LSTM's predictions in all six examples, we observe that it only predicts $ec4$. The Seq2Seq exhibits similar behavior but mostly predicts the lowest energy change phase ($ec0$). This example shows how LSTM and Seq2Seq fail to model flight phase transitions and usually predict only one flight phase type, no matter the length of the input or the initial flight phases exhibited. For these reasons, we conclude that the LSTM and Seq2Seq perform inadequately.

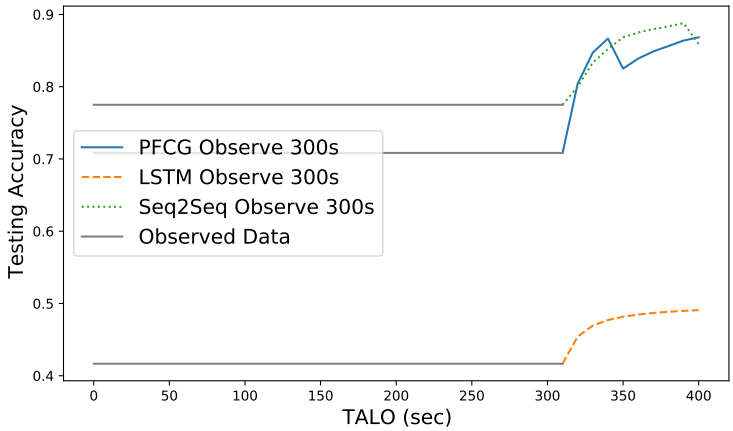
Now, let us consider a more complex example for flight phase prediction, as shown in Figure 7.12a. In this example, a HGV exhibits four different flight phases in increasing order: $ec1$ $ec2$ $ec3$ $ec4$. The methods observe 100 seconds of the trajectory and then predict 300 seconds of flight phases. The LSTM only predicts $ec4$ for this entire duration, demonstrating its inability to predict other flight phases. Thus, it fails in this example too. The Seq2Seq also does not perform well in this complicated scenario. It predicts $ec0$ for the entire prediction period. The proposed PCFG is the only method that correctly predicts the flight phase transitions, anticipating that a HGV will go through $ec1$ to $ec4$. However, the PCFG incorrectly adds a final phase transition pattern ($ec0$ $ec1$ $ec0$) at the end. These extra flight phases are due to the inaccurate phase duration estimations from the previous predictions. PCFG will accumulate error caused by inaccurate phase duration estimations, which contributes to the decrease in testing accuracy as time increases, as shown in the 100-second experiment in Figure 7.11a. This example indicates that although our PCFG predicts flight phase transitions well, there is still room for improvement.



(a) **Flight Phase Prediction Results after Observing 100s of a Trajectory.**



(b) **Flight Phase Prediction Results after Observing 200s of a Trajectory.**



(c) **Flight Phase Prediction Results after Observing 300s of a Trajectory.**

Figure 7.11. Flight Phase Prediction Accuracy as a Function of Time. We evaluate the proposed stochastic grammar PCFG, LSTM, and Seq2Seq with different lengths of observed flight phases as a function of time.

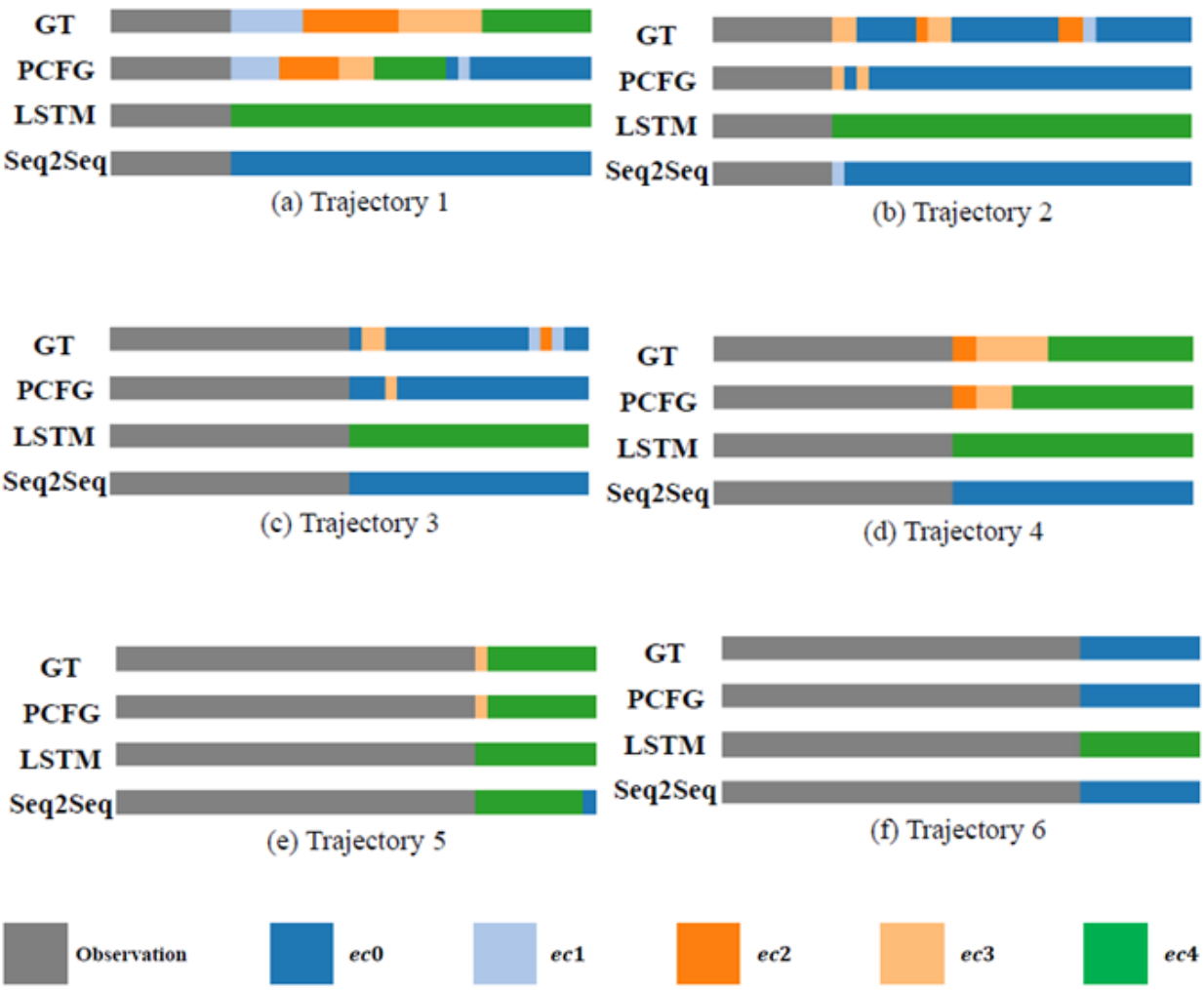


Figure 7.12. Flight Phase Prediction Results. We show the groundtruth (GT) flight phases (*i.e.*, the actual flight phases the testing trajectories exhibit during the prediction period) as well as prediction results with the proposed stochastic grammar PCFG, LSTM, and Seq2Seq for different observation lengths.

8. TRANSFER LEARNING FOR HYPERSONIC VEHICLE PREDICTION

8.1 Overview

In this chapter, we investigate transfer learning to predict the future behavior and engageability of hypersonic glide vehicles (HGVs). HGVs introduce new flight behaviors defined by high speeds (*i.e.*, Mach 5 to Mach 25) and high maneuverability [134]–[136], [138], [167], [213]. Their mobility allows them to evade traditional defense systems, so they pose a new set of challenges for monitoring their flight behavior [134], [135], [208], [213]. Due to their nascent development, there is limited HGV data available for use in designing HGV prediction methods. We propose a transfer learning approach to explore performance of our stochastic grammar prediction method on new, unseen trajectories and maneuvers. In this transfer learning scenario, a prediction method is derived using one dataset and evaluated on a second dataset. For added complexity, we train and evaluate our prediction method on datasets of HGV trajectories that exhibit different types of maneuvers. The first dataset contains trajectories that exhibit vertical maneuvers, which are behaviors related to changes in altitude. The second dataset exhibits both horizontal and vertical maneuvers, where horizontal maneuvers refer to changes in crossrange and downrange. The vertical dataset is also significantly smaller than the other dataset (*i.e.*, the dataset that exhibits horizontal and vertical maneuvers). We train our prediction approach on the smaller dataset of HGV trajectories that only exhibit vertical maneuvers. Then, we evaluate the HGV model obtained from the smaller dataset on a larger, more complicated HGV dataset that exhibits new behaviors never seen during development. With transfer learning, we investigate how our HGV prediction method extends to new, unseen HGV trajectories that might be encountered.

In this chapter, we use machine learning to derive a stochastic grammar for HGV prediction. A stochastic grammar is a rules-based framework used in sequential modeling tasks [184]–[186], [199]. Stochastic grammars have been used to predict words in sentences [184]–[186], [214]; envision RNA structures [187]–[190]; forecast human activities [191]–[193]; model human poses [194]; and detect anomalies in trajectory patterns [195]. A stochastic grammar consists of a set of production rules and a set of probabilities. The pro-

duction rules explicitly define how discrete categories transition in sequences. The probabilities describe how often each production rule (*i.e.*, each transition pattern) occurs. Stochastic grammars were initially developed for language modeling tasks, such as predicting words in sentences. In language applications, words are the discrete categories, and production rules describe how words transition throughout a sentence. For our application, the discrete categories are known as feature labels, which are quantized kinematic features based on altitude, velocity magnitude, and acceleration magnitude. The production rules describe how kinematic features transition throughout a trajectory. Our prior work indicates that stochastic grammars can succeed in modeling HGV behavior, even with limited training data [215]. In this chapter, we investigate their use for transfer learning to determine how well a stochastic grammar that describes vertical HGV maneuvers can perform on HGV trajectories that exhibit both horizontal and vertical maneuvers. We also propose a notional engageability definition to determine when a HGV could be intercepted. Along with future feature labels describing a HGV trajectory's maneuvers and behavior, we predict when a HGV is engageable. This investigation provides valuable information about how stochastic grammars capture physics-based information and can predict HGV behavior and engageability, even if they encounter new HGV maneuvers never seen during development.

8.2 Proposed Approach

Figure 8.1 shows an overview of our transfer learning approach. The analysis uses two different datasets of HGV trajectories. None of the trajectories in the two datasets are the same, and each trajectory is defined by altitude h; velocity magnitude $|v|$; and acceleration magnitude $|a|$ for each second of flight. Each dataset is used to train a separate stochastic grammar that models the HGV trajectories in that training dataset. The stochastic grammar derived from the first dataset is referred to as G_1, and the stochastic grammar derived from the second dataset is referred to as G_2. Next, each of the stochastic grammars is used to predict future values of HGV trajectories in the second dataset. The dotted line in Figure 8.1 indicates that only one stochastic grammar is used at a time to predict future HGV trajectory behavior. Whichever stochastic grammar (*i.e.*, G_1 or G_2) is used for prediction,

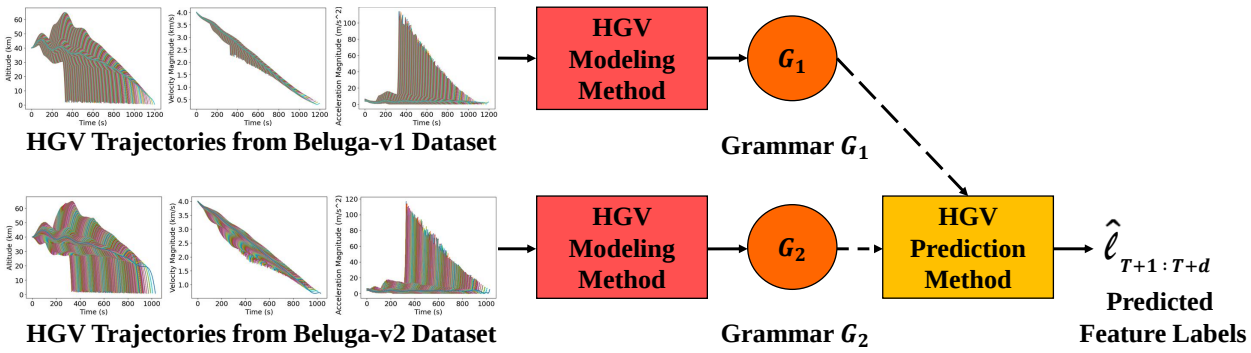


Figure 8.1. Transfer Learning Approach. Different datasets are used to derive two different stochastic grammars: G_1 and G_2. G_1 is derived from the first dataset of HGV trajectories (known as Beluga-v1), which exhibits vertical maneuvers. G_2 is derived from the second dataset of HGV trajectories (known as Beluga-v2), which exhibits both horizontal and vertical maneuvers. Then, each stochastic grammar is used to predict future HGV behavior of trajectories in the second dataset (*i.e.*, Beluga-v2). Note that only one stochastic grammar is used at a time – either G_1 or G_2 – for predictions, as indicated by the dotted lines in this figure. HGV predictions are feature labels $\hat{\ell}$ that represent different kinematic values. Predictions start at time $T+1$ and extend until time $T+d$. T refers to the observation duration in seconds, and d refers to the prediction duration in seconds. Larger versions of the six plots showing HGV trajectories in this figure are shown in Figures 8.2 and 8.3, when the datasets are explained in greater detail.

the output values of the prediction method form a sequence of d feature labels $\hat{\ell}_{T+1:T+d}$ that starts at time $T+1$ and ends at time $T+d$. T refers to the observation duration (*i.e.*, the period of time that a HGV trajectory is "observed" before making predictions), and d refers to the prediction duration (*i.e.*, the period of time that HGV trajectory behavior is predicted into the future). Both observation duration T and prediction duration d are measured in seconds. The predicted feature labels represent discrete categories corresponding to different ranges of kinematic features of the HGV trajectories. Recall that a stochastic grammar framework models discrete categories, such as words [184]. Thus, HGV trajectories (defined by continuous kinematic features) must be quantized into discrete categories to use a stochastic grammar to describe them. We refer to the discrete categories of HGV trajectories as "feature labels." The definition of the feature labels is provided in the next section.

8.2.1 Feature Labels

To produce the set of feature labels modeled by the stochastic grammar, each kinematic feature is quantized into I intervals. A quantizer is described by the thresholds (or intervals) that separate continuous inputs into discrete categories. The output values of the quantizer can be integer values that represent the specific quantization intervals [216]. We quantize each of the three kinematic features separately to obtain three sets of integer values that represent different ranges of each kinematic feature. Then, every combination of the quantization intervals (from each kinematic feature) is assigned a final integer value. This final integer value provides information about all three kinematic features of a HGV because it represents a combination of quantized kinematic features. We call these final integer values "feature labels", and a single feature label is represented as ℓ. HGV trajectories are quantized into feature labels before they are modeled by a stochastic grammar. Thus, our prediction approach predicts feature labels $\hat{\ell}$. We will now explain this process in greater detail.

Each kinematic feature $f \in \{h, |v|, |a|\}$ is divided into I intervals based on its distribution and range of kinematic feature values. Let M represent the number of HGV trajectories available for training our stochastic grammar. For the purposes of this example, let us assume that all trajectories fly for T seconds and thus have T data samples of feature f. Keep in mind that each trajectory flies for a different length of time in reality, though. For our example, the total number of data samples of a single kinematic feature from all trajectories is $N = MT$. The quantizer intervals are chosen such that $\frac{N}{I}$ data samples are included in each interval, and the quantized outputs Q are integers in the range $\{0, 1, ..., I-1\}$. Let F represent all values of f from all trajectories and time steps sorted in non-decreasing order: $F = \{f_0, f_1, ..., f_{N-1}\}$. Note that the minimum value of the feature is $f_{min} = f_0$, and the maximum value of the feature is $f_{max} = f_{N-1}$. The upper bound U of quantization interval i for kinematic feature f, where $i \in \{1, 2, ..., I\}$ is equal to the n-th kinematic feature value in F, where $n \in \{0, 1, ..., N-1\}$. In other words, $U_{f_i} = f_{n_i}$. The index n_i is computed as $\frac{\beta_i}{100}N$, where $\beta_i = \frac{100}{I}i$ represents the percentile that corresponds to the i-th interval. The calculation of n_i is:

$$n_i = \frac{\beta_i}{100}N = \frac{Ni}{I}. \tag{8.1}$$

Finally, the quantized output of a data sample of feature f at time t is:

$$Q_{f_t} = \begin{cases} 0, & f_t \in (-\infty, U_{f_1}], \\ i-1, & f_t \in (U_{f_{i-1}}, U_{f_i}] \; \forall \; i \neq 1, I, \\ I-1, & f_t \in (U_{f_{I-1}}, \infty]. \end{cases} \tag{8.2}$$

In practice, any number of quantization intervals I could be used. In this work, we use $I = 3$ so that the quantized outputs correspond to low, medium, and high values of a kinematic feature. With the numbering convention used, "0" corresponds to the lowest interval of a feature; "1" corresponds to the middle interval; and "2" corresponds to the highest interval. From the resulting quantization sets $\left(Q_h,\ Q_{|v|},\ \text{and}\ Q_{|a|}\right)$, we determine the set of all unique combinations of quantization intervals from each kinematic feature. We refer to this set as C since it is based on a Cartesian product of all quantization sets:

$$C = \{\psi_h \times \psi_{|v|} \times \psi_{|a|} \mid \psi_f \in Q_f \; \forall \; f \in \{h, |v|, |a|\}. \tag{8.3}$$

C contains I^3 elements, where "3" corresponds to the number of kinematic features used to define C. Thus, each element in C is a unique combination of quantization intervals from each kinematic feature. Finally, we assign each element in C an integer so that each unique combination has a unique label. Let this final set of integer feature labels be represented as $L = \{0, 1, ..., I^3 - 1\}$. In this work, there are twenty-seven unique combinations (and thus twenty-seven feature labels) based on $I = 3$ intervals and three kinematic features. Now that the set of feature labels has been defined, it can be used to label each trajectory. Let $h_{0:T}$, $|v|_{0:T}$, and $|a|_{0:T}$ represent a trajectory's altitude, velocity magnitude, and acceleration magnitude for T seconds of flight. Each feature f is quantized for every time $t \in 0{:}T$ using Equation 8.2. Then, each time t is assigned a feature label ℓ from L based on its specific combination of quantization levels from each kinematic feature. The final sequence of feature labels for the trajectory is represented as $\ell_{0:T}$. Next, stochastic grammars are derived to describe the transition patterns exhibited in these sequences of feature labels.

8.2.2 Engageability Definition

We are interested in moments during a trajectory where a HGV is engageable (*i.e.*, could be intercepted). To investigate a stochastic grammar's ability to predict HGV engageability, we define engageability based on the feature labels used in this chapter. For each second of flight $t \in 0{:}T$, a HGV could be considered *engageable* or *not engageable*. A HGV is considered engageable at time t if:

- altitude is medium or high ($Q_{h_t} \in \{1, 2\}$);
- velocity magnitude is low or medium $\left(Q_{|v|_t} \in \{0, 1\}\right)$; and
- acceleration magnitude is low $\left(Q_{|a|_t} \in \{0\}\right)$.

Based on this definition, only four of the twenty-seven feature labels are considered engageable. Each second of each trajectory is labeled based on these definitions. It should be noted that other definitions of engageability could be used, and our definition is notional [213].

8.2.3 HGV Modeling and Prediction

We use an unsupervised learning method known as automatic distillation of structure (ADIOS) [207] to derive the stochastic grammars. Recall that a stochastic grammar consists of a set of production rules R that describe transition patterns and a set of probabilities P that describe how often each rule occurs [184]–[186], [199], [214], [215]. ADIOS determines R and P by finding statistically significant transition patterns within the feature label sequences. It is an unsupervised learning method that learns R and P directly from the dataset, rather than relying on prior knowledge about the format or content of rules. ADIOS iterates multiple times over the training dataset, continuing to find new rules and combine rules, until R no longer changes. Next, we use the Generalized Earley Parser (GEP) [192], [193] for predicting feature labels. The GEP analyzes sequences of probability vectors, which contain a set of probabilities that indicate the likelihood that each feature label is exhibited. In this work, the probability vectors thus have a length of twenty-seven. GEP analyzes the probability vectors sequentially. First, it examines the feature label transition patterns

exhibited during an initial, "observed" portion of a trajectory. Then, it determines the set of all rules that match the observed portion of flight. GEP constructs a set of all possible feature label sequences that could be exhibited during the rest of the trajectory's flight by using the rules in R. Finally, GEP selects the most probable sequence of feature labels based on the probabilities in P. We have used ADIOS and GEP for HGV modeling and prediction before. More information about these approaches are included in our prior work [215].

8.3 Experimental Setup

8.3.1 Datasets

We utilize two datasets of HGV trajectories to develop and evaluate our prediction method. Both datasets were generated by a publicly available tool known as Beluga [217], [218]. Beluga is a general-purpose indirect trajectory optimization framework. It generates HGV trajectories with the aerodynamic profile of [218], [219] according to flight mechanics described in [220]. Beluga simulates HGV trajectories after the HGV enters the atmosphere. All trajectories adhere to a re-entry model where only gravity, lift, and drag operate on the vehicle. The first dataset (known as Beluga-v1) consists of 250 HGV trajectories. These trajectories are designed to exhibit vertical maneuvers, which are maneuvers in terms of altitude. The second dataset (known as Beluga-v2) consists of 1,681 trajectories. These trajectories exhibit both vertical maneuvers and horizontal maneuvers simultaneously. Horizontal maneuvers refer to maneuvers in terms of crossrange and downrange. Because each dataset contains trajectories that exhibit distinct behaviors, we can use them in a transfer learning scenario to investigate the impact of using only one of the datasets to prepare a prediction method for success on the other dataset.

Figures 8.2 and 8.3 show the kinematic features h, $|v|$, and $|a|$ for all HGV trajectories from both datasets. Although the trajectories in each dataset exhibit different types of maneuvers, these figures show how their kinematic features look relatively similar. From a visual perspective, these plots indicate that transfer learning could succeed in this scenario because the kinematic features are not significantly different from each other. A numerical analysis supports this qualitative analysis. Table 8.1 includes information about each

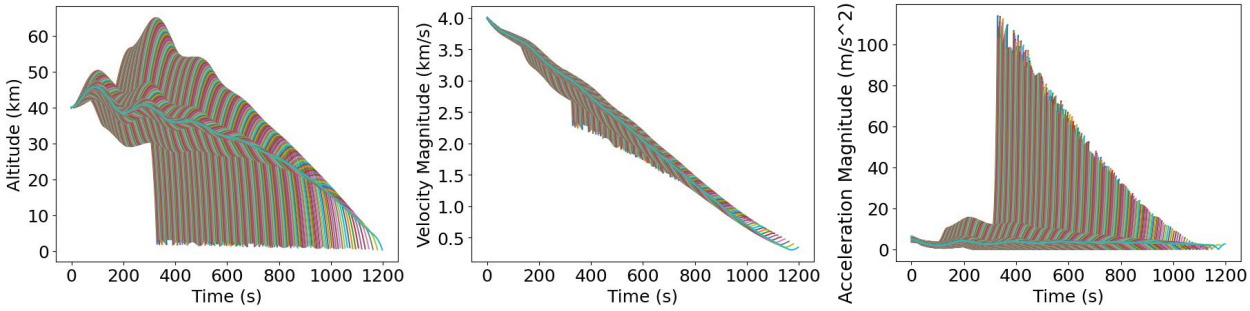


Figure 8.2. Kinematic Features of HGV Trajectories in the Beluga-v1 Dataset. Colors are randomly assigned to each trajectory and carry no significance. They help show individual trajectories.

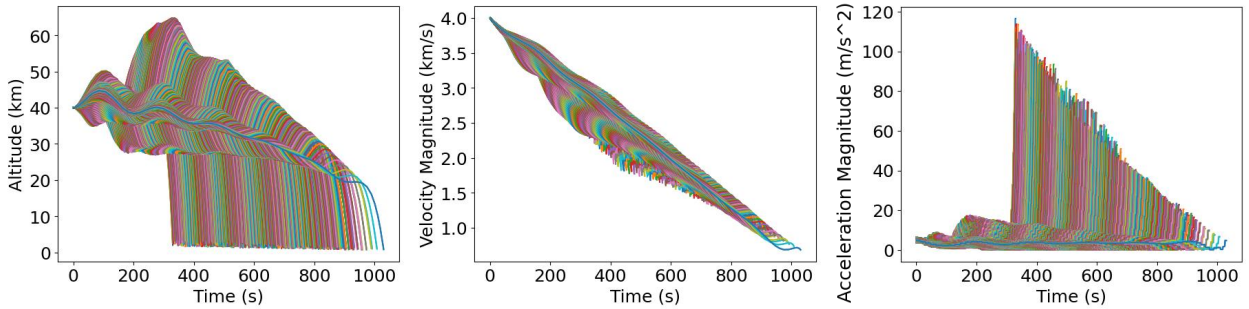


Figure 8.3. Kinematic Features of HGV Trajectories in the Beluga-v2 Dataset. Colors are randomly assigned to each trajectory and carry no significance. They help show individual trajectories.

quantization bound from both datasets. Recall that these bounds are determined directly from the training data based on the quantization process described in Section 8.2.1. Because we quantize each kinematic feature into three intervals using a percentile-based approach, each bound separates 33.33% of data samples (*i.e.*, kinematic values from all trajectories and time steps) into different intervals of that kinematic feature. Any number of quantization intervals/bounds could be used in practice, but we use three intervals in this chapter to represent low, medium, and high ranges of values for each kinematic feature. From the information in Table 8.1, notice that Upper Bound 1 U_{h_1} for the Beluga-v1 dataset is 36,201 m, which indicates that 33.33% of altitude data samples from the Beluga-v1 dataset are less than 36,201 m. This same bound for the Beluga-v2 dataset is 37,035 m. Thus, even though each dataset exhibits different behaviors, their summary statistics are relatively similar and produce bounds that are fairly close to each other. If the data distributions were drastically

Table 8.1. Quantization Bounds from Each HGV Dataset. Upper Bound 3 is not included in this table for any kinematic feature since $U_{f_3} = \infty \; \forall \; f \in \{h, |v|, |a|\}$.

Quantization Bounds from Each HGV Dataset						
Dataset	Altitude (m)		Velocity Magnitude (m/s)		Acceleration Magnitude (m/s^2)	
	Upper Bound 1 U_{h_1}	Upper Bound 2 U_{h_2}	Upper Bound 1 $U_{\lvert v\rvert_1}$	Upper Bound 2 $U_{\lvert v\rvert_2}$	Upper Bound 1 $U_{\lvert a\rvert_1}$	Upper Bound 2 $U_{\lvert a\rvert_2}$
Beluga-v1	36,201	42,016	2,575	3,317	2.3	3.6
Beluga-v2	37,035	41,927	2,579	3,316	2.3	3.8

different from each other, the quantization bounds obtained based on each dataset would also be very different, and a transfer learning approach might not succeed. However, our numerical analysis shows that it is possible to succeed with transfer learning because the distributions of kinematic features in the HGV datasets are not significantly different, even if their underlying maneuvers are.

Table 8.2. Number of HGV Trajectories in Each Beluga Dataset. We explore a limited training data scenario, using only 20% of trajectories in each dataset as training data. Notice that there are significantly fewer trajectories in Beluga-v1 compared to Beluga-v2.

HGV Datasets Splits			
HGV Dataset	Training Trajectories	Testing Trajectories	Total Trajectories
Beluga-v1	50	200	250
Beluga-v2	336	1,345	1,681

Now that we have explored the HGV datasets used in our analysis, we will present our experimental setup. For our experiments, we use 20% of trajectories as training data and 80% of trajectories as testing data. This means that only 50 Beluga-v1 trajectories are used to derive stochastic grammar G_1, and 336 Beluga-v2 trajectories are used to derive stochastic

grammar G_2. Then, the HGV prediction method is evaluated on the remaining 200 Beluga-v1 and 1,345 Beluga-v2 trajectories. The reason we split our data in this fashion is to explore prediction results in a limited data scenario. We develop our approach under the assumption that a large amount of actual HGV trajectories might not be available for training. Table 8.2 shows the summary of the training and testing trajectories for each dataset.

8.3.2 Evaluation Metrics

Recall that $\hat{\ell}_{T+1:T+d}$ represents the predicted feature labels of a HGV trajectory, where T represents the total number of seconds that are observed and used to predict d seconds into the future. We compute accuracy, weighted precision, weighted recall, and weighted F1 [93] of the predicted feature labels $\hat{\ell}$ by comparing them to the actual, true feature labels ℓ. Let M represent the number of trajectories in the test set. We compute accuracy as:

$$\text{Accuracy} = \frac{1}{Md} \sum_{j=0}^{M-1} \sum_{k=1}^{d} \mathbb{1}(\hat{\ell}_{j,T+k}, \ell_{j,T+k}), \tag{8.4}$$

where $\hat{\ell}_{j,T+k}$ is the predicted feature label of trajectory j at time $T+k$; $\ell_{j,T+k}$ is the corresponding true feature label; and $\mathbb{1}(\cdot)$ is the indicator function. Accuracy is a measure of how many feature labels were predicted correctly. Weighted precision is computed as:

$$\text{Weighted Precision} = \frac{1}{\sum_{l \in L} |\ell_l|} \sum_{l \in L} |\ell_l| \frac{|\ell_l \cap \hat{\ell}_l|}{|\hat{\ell}_l|}, \tag{8.5}$$

where ℓ represents all true feature labels (from all trajectories and predicted time steps) that are being evaluated; $\hat{\ell}$ represents all corresponding predicted feature labels (from all trajectories and predicted time steps); L is the set of unique feature labels defined in Section 8.2.1 (*i.e.*, the set of "words" modeled by the stochastic grammar); ℓ_l is the subset of ℓ with the specific feature label $l \in L$; and $\hat{\ell}_l$ is the subset of $\hat{\ell}$ with feature label l. Precision is a measure of our prediction method's ability not to predict samples as a certain feature label when that particular feature label is being evaluated. Recall measures the prediction method's

ability to predict a particular feature label when that feature label is actually exhibited by a trajectory. Weighted recall is then:

$$\text{Weighted Recall} = \frac{1}{\sum_{l \in L} |\ell_l|} \sum_{l \in L} |\ell_l| \frac{|\ell_l \cap \hat{\ell}_l|}{|\ell_l|}, \tag{8.6}$$

Our final evaluation metric is weighted F1, defined as:

$$\text{Weighted F1} = \frac{1}{\sum_{l \in L} |\ell_l|} \sum_{l \in L} |\ell_l| \frac{\frac{|\ell_l \cap \hat{\ell}_l|}{|\hat{\ell}_l|} * \frac{|\ell_l \cap \hat{\ell}_l|}{|\ell_l|}}{\frac{|\ell_l \cap \hat{\ell}_l|}{|\hat{\ell}_l|} + \frac{|\ell_l \cap \hat{\ell}_l|}{|\ell_l|}}. \tag{8.7}$$

It is a measure of the balance between precision and recall. It can be interpreted as a weighted harmonic mean between the two. Note that we calculate weighted precision, weighted recall, and weighted F1. Weighted metrics reflect the frequency of each feature label's occurrence in the evaluation dataset. For all of these metrics, larger values are better, and 100% is the best possible score.

8.3.3 Experimental Results

Table 8.3. Results Predicting Feature Labels. The top two rows in the table show results obtained by training and testing on the same dataset. The third row shows the results from our transfer learning experiment.

Feature Label Prediction Results						
Training Dataset	Testing Dataset	Accuracy	Weighted Precision	Weighted Recall	Weighted F1	Number of $\boldsymbol{G}$ Rules
Beluga-v1	Beluga-v1	47.55%	60.93%	47.55%	49.73%	48
Beluga-v2	Beluga-v2	35.80%	52.49%	35.80%	39.45%	265
Beluga-v1	Beluga-v2	39.22%	48.54%	39.22%	41.31%	48

Tables 8.3 and 8.4 show results of observing HGV trajectories for 300 seconds (*i.e.*, $T = 300$) and predicting feature labels and engageability, respectively, for 100 seconds (*i.e.*, $d = 100$). In both cases, we first train and evaluate our prediction method on the same dataset. For example, we train a stochastic grammar on Beluga-v1 and use it to predict trajectories in

Table 8.4. Results Predicting Engageability. The top two rows in the table show results obtained by training and testing on the same dataset. The third row shows the results from our transfer learning experiment.

Engageability Prediction Results						
Training Dataset	Testing Dataset	Accuracy	Weighted Precision	Weighted Recall	Weighted F1	Number of $\boldsymbol{G}$ Rules
Beluga-v1	Beluga-v1	73.75%	81.42%	73.75%	74.79%	48
Beluga-v2	Beluga-v2	67.14%	83.08%	67.14%	69.01%	265
Beluga-v1	Beluga-v2	69.09%	82.77%	69.09%	71.92%	48

Beluga-v1. In this case, we obtain the highest metrics for feature label prediction, such as a prediction accuracy of 47.55%. This process is repeated for the Beluga-v2 dataset to obtain a prediction accuracy of 35.80%. Recall that the Beluga-v2 dataset is more complicated than the Beluga-v1 dataset because Beluga-v2 exhibits maneuvers in both the horizontal and vertical directions, whereas Beluga-v1 only exhibits maneuvers in the vertical direction. Due to the increased complexity of Beluga-v2, our prediction method's performance drops. Next, we obtain results with transfer learning. We use G_1 (*i.e.*, the stochastic grammar derived from the Beluga-v1 dataset) to predict future feature labels and engageability of trajectories in the Beluga-v2 dataset. With this transfer learning approach, the prediction performance on the Beluga-v2 dataset improves, achieving a feature label prediction accuracy of 39.22%. Results indicate that transfer learning can be used to not only maintain performance on a more complicated dataset but to also improve results slightly. For example, feature label prediction accuracy on the Beluga-v2 dataset improves from 35.80% to 39.22% with transfer learning.

Similar results are observed with engageability predictions. Stochastic grammar G_1 trained and evaluated on the Beluga-v1 dataset achieves the highest engageability metrics. Stochastic grammar G_2 trained and evaluated on the Beluga-v2 dataset exhibits a drop in engageability prediction metrics. However, using G_1 on the Beluga-v2 dataset improves all engageability prediction metrics. For example, engageability prediction accuracy increases from 67.14% to 69.09% when switching from G_2 to G_1 on the Beluga-v2 dataset. In gen-

eral, all engageability prediction metrics are higher than feature label prediction metrics. This is because feature label prediction requires explicitness. The exact feature label (out of twenty-seven different feature labels) must be predicted in order to be considered correct, which makes this task more difficult. For engageability prediction, the exact feature label does not need to be predicted. Instead, a feature label that has the same engageability classification (*i.e.*, *engageable* or *not engageable*) could be predicted, and the prediction is still considered correct. Furthermore, there are only two different prediction outputs for enageability, compared to the twenty-seven for feature label prediction. Although this is a slightly less difficult evaluation procedure, it still provides very useful information about the status of a HGV. Because there could be an unequal number of *engageable* vs. *not engageable* feature labels, we need to evaluate weighted precision, weighted recall, and weighted F1 to understand how our prediction method performs in spite of the imbalance. However, we see that all engageability metrics are consistent with each other and very high, which indicates that our prediction approach can perform well despite any class imbalances that may exist.

Our transfer learning experiments demonstrate a stochastic grammar learned from a smaller, less complicated dataset can still succeed on a larger, more complicated dataset. This is important since it is difficult to obtain HGV trajectory data, so all HGV behaviors might not be available in a training dataset. As seen in Tables 8.3 and 8.4, G_1 contains 48 rules, while G_2 contains 265 rules. G_2 contains significantly more rules than G_1, yet G_1 predicts HGV feature labels and engageability better on the Beluga-v2 dataset. These results indicate that G_1 contains rules that capture physics-based information better than the rules in G_2, thus allowing G_1 to understand Beluga-v2 trajectories even though their horizontal maneuvers were not used to train G_1. From these experiments, we conclude that G_2 contains more rules that model more unique, specific HGV behaviors. In other words, they capture "edge cases" that are not often exhibited and thus do not generalize as well to new trajectories. In conclusion, not all of the rules in G_2 are necessary to predict HGV trajectories well. This phenomenon is often observed with stochastic grammar-based approaches [184]–[186], [214]. In general, stochastic grammars with more rules have more ambiguity, meaning that there are more rules that match observed input sequences. Because there are more rules that match the input, there are more options that the GEP can select

in order to predict the rest of the sequence. This means that the chances of choosing an incorrect rule to predict the next sequence value are higher. Therefore, stochastic grammars with fewer rules that are more descriptive often provide better prediction results. With these types of stochastic grammars, there are fewer rules to choose from, less ambiguity, and more chances of choosing the correct rule for prediction. One other benefit of a stochastic grammar with fewer rules is that prediction times are shorter. There are fewer rules for a parser to iterate through and keep track of, which makes computation faster. Our transfer learning approach demonstrates all of these benefits to achieve better performance on a larger, more complicated HGV dataset, even when the trajectories in the new dataset exhibit new behaviors.

9. SUMMARY AND FUTURE WORK

In Chapter 2, we propose a CNN approach to analyze audio signal spectrograms for the purpose of validating the audio signal authenticity. The experimental results show that the method accomplishes this discrimination task with high accuracy on the test dataset with a relatively shallow network. Our method generalizes to new audio attacks never seen during training. Thus, our results indicate that a signals-informed and signals-based approach assists a neural network in learning information to extend to new attacks. However, our method fails to classify other new audio signals correctly. Future work should focus on understanding the failure cases and improving our method to correctly identify whether they are fake or real audio signals. A future approach could include analyzing the signals with a Natural Language Processing (NLP) approach to evaluate the coherence of the spoken phrases. Then, two analyses could be conducted in parallel to analyze the frequency content and structure of the signal as well as the coherence of the spoken words. Another future direction could include an environmental analysis of the captured audio signal. If, for example, an audio signal is identified to be recorded outside but the speaker says phrases as if he or she is indoors, this mismatch between recording environment and spoken cues could indicate that the audio is synthesized. These experiments conducted in tandem with our proposed approach would strengthen our audio authentication method.

In Chapter 3, we improve upon our previous work to demonstrate the benefits of using higher-resolution spectrograms and an attention mechanism. We demonstrate that a neural network that utilizes both convolution and transformer capabilities achieves high success in detecting synthesized speech. Convolution operations convert a spectrogram input image into feature maps that contain salient information for discriminating synthesized and genuine human speech. It also enables a transformer to achieve high success with less data than transformers typically require. We demonstrate that this synthesized speech detection method performs well on a publicly available dataset and also that it generalizes to lossy audio signals in a sequestered dataset of an unknown nature.

Although this approach is promising, future work should consider more diverse speech features. For example, the method should be validated on data of different audio formats,

compression levels, sampling rates, and durations. Since both the CNN and the CCT perform well, an ensemble of these two methods could be created and augmented with other neural networks. Finally, a speech analysis method such as this could be paired with methods that analyze media's other data modalities. For example, our synthesized speech detector could analyze speech signals found in videos, while methods that analyze images and videos could analyze the visual content. A metadata analysis could strengthen this multi-modal approach even more.

In Chapter 4, we investigate three transformers for synthesized speech detection and proposes a transformer ensemble to boost performance. We show that our proposed transformer ensemble achieves better synthesized speech detection than each of the individual transformers, especially considering the highly imbalanced nature of our experimental dataset. We also demonstrate that our transformer ensemble can achieve the same level of high success, even when analyzing shorter speech signals. Future work will focus on other tasks in addition to detection, such as localization of synthesized speech within full audio signals.

Finally, we conclude our audio analysis work in Chapter 5. In this chapter, we attribute speech signals to speech synthesizers in both closed set and open set scenarios. We analyze speech signals in the form of spectrograms with a compact attribution transformer (CAT) and demonstrate that our approach achieves success on this speech synthesizer attribution task. Furthermore, we show that poly-1 loss formulations improve results and that analyzing the latent space of CAT with tSNE can discriminate between different unknown synthesizers. Future work could extend this analysis to more speech synthesizers and increase separability in the latent space.

In Chapter 6, we commence our discussion of machine learning for hypersonic vehicle applications. We utilize machine learning methods for HGV trajectory classification. Our techniques exploit a data driven approach, learning how to distinguish HGVs and CRVs directly from the available training data. Our approaches analyze the aerodynamic properties of the vehicles to learn a unique signature that describes each one. Experimental results shows that our approach works well and increases in performance over TALO.

Future research directions could explore how these methods extend to trajectories of other vehicles. It is important to consider both different types of HGVs and CRVs as well as

entirely new vehicles. Additionally, it is necessary to develop an approach to consider out-of-distribution vehicles. In other words, the methods should be able to identify a new type of vehicle never seen before during training. These are all future directions for improving our approach.

In Chapter 7, we propose a stochastic grammar model for flight phase prediction. We demonstrate that the proposed approach successfully predicts future flight phases of HGVs and estimates temporal durations of the flight phases, even with limited training data. Although our approach performs well, there is still room for improvement. Future work could extend our analysis to include different types of HGV vehicles. We currently simulate all trajectories with the same aerodynamic profile, so it would be useful to investigate how our approach performs when multiple aerodynamic profiles are used. The set of flight phase transition rules would be more complicated, as different vehicles would transition between phases differently. Another direction for future work will focus on improving the flight phase duration priors. Our method predicts flight phase transitions well, but it could be improved in terms of its duration estimation by considering the information from the current observations.

Finally in Chapter 8, we propose transfer learning for HGV trajectory prediction. We analyze three kinematic features (altitude, velocity magnitude, and acceleration magnitude) to predict future kinematic features and engageability of HGV trajectories. To do so, we derive stochastic grammars from two different datasets that are different sizes and exhibit different behaviors. Results indicate that a stochastic grammar trained on the smaller dataset can still perform well (and even increase prediction results) on the larger dataset, even though the larger dataset exhibits new types of HGV behavior. This investigation shows that transfer learning can be used to predict HGV trajectories in limited training data scenarios. As with any machine learning or deep learning approach, it can be helpful to have an explanation for prediction results. In the future, future work could investigate the explainability of various machine learning methods to understand their decision-making better.

9.1 Contributions of this Dissertation

The contributions of this dissertation are as follows:

- we investigate a CNN and multiple transformers for synthesized speech detection and demonstrate success on this task;
- we show that our transformer-based synthesized speech detection approach also succeeds on a sequestered dataset of lossy audio signals, which indicates that it is able to generalize to new, unseen data samples not present in the training dataset;
- we design a transformer ensemble to improve synthesized speech detection;
- we demonstrate that our transformer ensemble can achieve high success on this task, even when analyzing shorter portions (*i.e.*, less) of a speech signal;
- we propose a compact attribution transformer (CAT) to attribute synthesized speech signals to known and unknown speech synthesizers;
- we investigate the latent space of CAT with tSNE to distinguish between different speech synthesizers and offer more insight into the trained neural network;
- we demonstrate that CAT successfully discriminates between different known and unknown synthesizers, which indicates that CAT generalizes to new speech synthesizers not seen during training;
- we use poly-1 loss formulations to improve attribution results;
- we investigate a CNN for vehicle classification based on initial phases of a vehicle's flight;
- we demonstrate that machine learning approaches successfully discriminate between three types of vehicles (two different hypersonic vehicles and one conic re-entry vehicle), even under noisy conditions;
- we introduce stochastic grammars to model HGV trajectories;

- we demonstrate that our stochastic grammar method accurately predicts HGV flight phases, even with a limited amount of training data;
- we show that our vehicle classification and HGV prediction methods achieve higher success as time after lift-off (TALO) increases;
- we propose an engageability definition that indicates when a HGV could be intercepted;
- we use transfer learning to derive a stochastic grammar with a smaller dataset of HGV trajectories that exhibit vertical maneuvers and predict future behavior and engageability of HGV trajectories in a larger dataset that exhibit both horizontal and vertical maneuvers;
- we demonstrate that transfer learning with stochastic grammars can predict HGV behavior and engageability, even when the method encounters HGV trajectories that exhibit different maneuvers than the trajectories used to derive the stochastic grammar.

9.2 Publications

Journal Publications Resulting from this Dissertation:

1. **E. R. Bartusiak**, M. A. Jacobs, M. W. Chan, M. L. Comer, and E. J. Delp, "Predicting Hypersonic Glide Vehicle Behavior with Stochastic Grammars", in *IEEE Transactions on Aerospace and Electronics*, 2023, *Submitted for Review.*

Book Chapters Resulting from this Dissertation:

2. H. Hao*, **E. R. Bartusiak***, D. Güera, D. Mas, S. Baireddy, Z. Xiang, S. K. Yarlagadda, R. Shao, J. Horváth, J. Yang, F. Zhu, and E. J. Delp, "Deepfake Detection Using Multiple Data Modalities", in *Handbook of Digital Face Manipulation and Detection - From DeepFakes to Morphing Attacks, Series on Advances in Computer Vision and Pattern Recognition*, Springer, Cham, January 2022, pp. 235-254. DOI: 10.1007/978-3-030-87664-7_11.

Conference Publications Resulting from this Dissertation:

3. **E. R. Bartusiak**, K. Bhagtani, A. K. S. Yadav, and E. J. Delp, "Transformer Ensemble for Synthesized Speech Detection", 2023, *Submitted for Review.*

4. **E. R. Bartusiak**, M. A. Jacobs, C. F. Spells, M. W. Chan, M. L. Comer, and E. J. Delp, "Transfer Learning for Hypersonic Vehicle Trajectory Prediction", *Proceedings of the IEEE Aerospace Conference*, pp. 1-8, March 2023, Big Sky, MT, USA.

5. **E. R. Bartusiak** and E. J. Delp, "Transformer-Based Speech Synthesizer Attribution in an Open Set Scenario", *Proceedings of the IEEE International Conference on Machine Learning and Applications*, pp. 1-8, December 2022, Nassau, The Bahamas. DOI: 10.48550/arXiv.2210.07546.

*denotes equal authorship.

6. **E. R. Bartusiak**, H. Hao, M. A. Jacobs, N. X. Nguyen, M. W. Chan, M. L. Comer, and E. J. Delp, "A Stochastic Grammar Approach to Predict Flight Phases of a Hypersonic Glide Vehicle", *Proceedings of the IEEE Aerospace Conference*, pp. 1-15, March 2022, Big Sky, MT, USA. DOI: 10.1109/AERO53065.2022.9843362.

7. **E. R. Bartusiak** and E. J. Delp, "Synthesized Speech Detection Using Convolutional Transformer-Based Spectrogram Analysis", *Proceedings of the IEEE Asilomar Conference on Signals, Systems, and Computers*, pp. 1426-1430, October 2021, Asilomar, CA, USA (Virtual). DOI: 10.1109/IEEECONF53345.2021.9723142.

8. **E. R. Bartusiak**, N. X. Nguyen, M. W. Chan, M. L. Comer, and E. J. Delp, "A Machine Learning Approach to Classify Hypersonic Vehicle Trajectories", *Proceedings of the IEEE Aerospace Conference*, pp. 1-14, March 2021, Big Sky, MT, USA (Virtual). DOI: 10.1109/AERO50100.2021.9438274.

9. **E. R. Bartusiak** and E. J. Delp, "Frequency Domain-Based Detection of Generated Audio", *Proceedings of the Media Watermarking, Security, and Forensics Conference, IS&T Electronic Imaging Symposium*, pp. 273(1)-273(7), January 2021, Burlingame, CA, USA (Virtual). DOI: 10.48550/arXiv.2205.01806.

Other Publications:

10. K. Bhagtani, **E. R. Bartusiak**, A. K. S. Yadav, P. Bestagini, and E. J. Delp, "Synthesized Speech Attribution Using the Patchout Spectrogram Attribtion Transformer", 2023, *Submitted for Review.*

11. A. K. S. Yadav, Z. Xiang, **E. R. Bartusiak**, P. Bestagini, S. Tubaro, and E. J. Delp, "ASSD: Synthetic Speech Detection in the AAC Compressed Domain", 2023, *Submitted for Review.*

12. A. K. S. Yadav, **E. R. Bartusiak**, K. Bhagtani, and E. J. Delp, "Synthetic Speech Attribution Using Self Supervised Audio Spectrogram Transformer", *Proceedings of the*

Media Watermarking, Security, and Forensics Conference, IS&T Electronic Imaging Symposium, January 2023, San Francisco, CA, USA.

13. **E. R. Bartusiak**, M. Barrabés, A. Rymbekova, J. Gimbernat-Mayol, C. Lopéz, L. Barberis, D. Mas, X. Giró-i-Nieto, and A. G. Ioannidis, "Predicting Dog Phenotypes from Genotypes", *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 3558-3562, July 2022, Glasgow, Scotland, UK. DOI: 10.1109/EMBC48229.2022.9870905.

14. K. Bhagtani, A. K. S. Yadav, **E. R. Bartusiak**, Z. Xiang, R. Shao, S. Baireddy, and E. J. Delp, "An Overview of Recent Work in Multimedia Forensics: Methods and Threats", pp. 1-17, November 2022, *arXiv:2204.12067*. DOI: 10.48550/arXiv.2204.12067.

15. K. Bhagtani, A. K. S. Yadav, **E. R. Bartusiak**, Z. Xiang, R. Shao, S. Baireddy, and E. J. Delp, "An Overview of Recent Work in Multimedia Forensics", *Proceedings of the IEEE International Conference on Multimedia Information Processing and Retrieval*, pp. 324-329, August 2022, San Jose, CA, USA (Virtual). DOI: 10.1109/MIPR54900.2022.00064.

16. E. D. Cannas, S. Baireddy, **E. R. Bartusiak**, S. K. Yarlagadda, D. Mas, P. Bestagini, S. Tubaro, and E. J. Delp, "Open-Set Source Attribution for Panchromatic Satellite Imagery", *Proceedings of the IEEE International Conference on Image Processing*, pp. 3038-3042, September 2021, Anchorage, AK, USA (Virtual). DOI: 10.1109/ICIP42928.2021.9506600.

17. H. Hao, S. Baireddy, **E. R. Bartusiak**, L. Konz, K. LaTourette, M. Gribbons, M. Chan, M. Comer, and E. J. Delp, "An Attention-Based System for Damage Assessment Using Satellite Imagery", *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, pp. 4396-4399, July 2021, Brussels, Belgium (Virtual). DOI: 10.1109/IGARSS47720.2021.9554054.

18. H. Hao, S. Baireddy, **E. R. Bartusiak**, M. Gupta, K. LaTourette, L. Konz, M. Gribbons, M. Chan, M. Comer, and E. J. Delp, "Building Height Estimation via Satellite

Metadata and Shadow Instance Detection", *Proceedings of the SPIE Automatic Target Recognition Conference*, pp. 1-16, April 2021, Orlando, FL, USA (Virtual). DOI: 10.1117/12.2585012.

19. D. Mas, H. Hao, S. K. Yarlagadda, S. Baireddy, R. Shao, J. Horváth, J. Yang, **E. R. Bartusiak**, D. Güera, F. Zhu, and E. J. Delp, "Deepfakes Detection with Automatic Face Weighting", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Workshop on Media Forensics*, pp. 2851-2859, June 2020, Seattle, WA, USA (Virtual). DOI: 10.1109/CVPRW50498.2020.00342.

20. **E. R. Bartusiak**, "An Adversarial Approach to Spliced Forgery Detection and Localization in Satellite Imagery", *Purdue University*, pp. 1-54, May 2019, West Lafayette, IN, USA. DOI: 10.25394/PGS.8035952.v1.

21. **E. R. Bartusiak**, S. K. Yarlagadda, D. Güera, F. Zhu, P. Bestagini, S. Tubaro, and E. J. Delp, "Splicing Detection And Localization In Satellite Imagery Using Conditional GANs", *Proceedings of the IEEE International Conference on Multimedia Information Processing and Retrieval* , pp. 91-96, March 2019, San Jose, CA, USA. DOI: 10.1109/MIPR.2019.00024.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, pp. 436–444, May 2015. DOI: 10.1038/nature14539.

[2] F. Rosenblatt, "The Perceptron: A Perceiving and Recognizing Automation," *Cornell Aeronautical Laboratory*, pp. 1–33, Jan. 1957. [Online]. Available: https://blogs.umass.edu/brain-wars/files/2016/03/rosenblatt-1957.pdf.

[3] O. G. Selfridge, "Pandemonium: A Paradigm for Learning," *Proceedings of the Symposium on Mechanisation of Thought Processes*, pp. 511–529, 1959. [Online]. Available: https://aitopics.org/download/classics:504E1BAC.

[4] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, Jul. 1959. DOI: 10.1147/rd.33.0210.

[5] K. Hao, "What is Machine Learning?" *MIT Technology Review*, Nov. 2018. [Online]. Available: https://www.technologyreview.com/2018/11/17/103781/what-is-machine-learning-we-drew-you-another-flowchart/.

[6] P. Werbos, "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences," *PhD Thesis, Harvard University*, pp. 1–454, Aug. 1974, Cambridge, MA, USA. [Online]. Available: https://www.researchgate.net/profile/Paul-Werbos/publication/35657389_Beyond_regression_new_tools_for_prediction_and_analysis_in_the_behavioral_sciences/links/576ac78508aef2a864d20964/Beyond-regression-new-tools-for-prediction-and-analysis-in-the-behavioral-sciences.pdf.

[7] D. B. Parker, "Learning Logic: Casting the Cortex of the Human Brain in Silicon," *Center for Computational Research in Economics and Management Science*, pp. 1–47, 1985.

[8] Y. Lecun, "Une procedure d'apprentissage pour reseau a seuil asymmetrique (A Learning Scheme for Asymmetric Threshold Networks)," *Proceedings of Cognitiva 85*, pp. 599–604, 1985, Paris, France. [Online]. Available: http://yann.lecun.com/exdb/publis/pdf/lecun-85.pdf.

[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-Propagating Errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986. DOI: 10.1038/323533a0.

[10] B. E. Boser, E. Sackinger, J. Bromley, Y. Le Cun, and L. D. Jackel, "An Analog Neural Network Processor with Programmable Topology," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 12, pp. 2017–2025, Dec. 1991. DOI: 10.1109/4.104196.

[11] R. Raina, A. Madhavan, and A. Y. Ng, "Large-Scale Deep Unsupervised Learning using Graphics Processors," *Proceedings of the International Conference on Machine Learning*, pp. 873–880, Jun. 2009, Montréal, Canada. DOI: 10.1145/1553374.1553486.

[12] C. Farabet, Y. LeCun, K. Kavukcuoglu, *et al.*, "Scaling Up Machine Learning: Parallel and Distributed Approaches," *Proceedings of the ACM SIGKDD International Conference Tutorials*, pp. 399–419, Jan. 2011, San Diego, CA, USA. DOI: 10.1145/2107736.2107740.

[13] G. E. Hinton, S. Osindero, and Y. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006. DOI: 10.1162/neco.2006.18.7.1527.

[14] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy Layerwise Training of Deep Networks," *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 153–160, Dec. 2006, Vancouver, Canada. [Online]. Available: https://dl.acm.org/citation.cfm?id=2976476.

[15] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun, "Efficient Learning of Sparse Representations with an Energy-Based Model," *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 1137–1144, Dec. 2006, Vancouver, Canada. [Online]. Available: https://dl.acm.org/citation.cfm?id=2976599.

[16] D. C. Cireçan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, Big, Simple Neural Nets for Handwritten Digit Recognition," *Neural Computation*, vol. 22, pp. 3207–3220, 12 Dec. 2010. DOI: 10.1162/NECO_a_00052.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 1–9, Dec. 2012, Lake Tahoe, NV, USA. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[18] R. Kohavi and F. Provost, "Glossary of Terms: Special Issue on the Applications of Machine Learning and the Knowledge Discovery Process," *Machine Learning*, vol. 30, pp. 271–274, 1998. [Online]. Available: https://ai.stanford.edu/~ronnyk/glossary.html.

[19] S. Malodia, N. Islam, P. Kaur, and A. Dhir, "Why Do People Use Artificial Intelligence (AI)-Enabled Voice Assistants?" *IEEE Transactions on Engineering Management*, pp. 1–15, Dec. 2021. DOI: 10.1109/TEM.2021.3117884.

[20] Apple, "Siri Does More than Ever. Even Before You Ask.," *Apple.com*, 2021. [Online]. Available: https://www.apple.com/siri/.

[21] Apple, "Use Siri in Your Car," *Apple.com*, 2022. [Online]. Available: https://support.apple.com/guide/iphone/use-siri-in-your-car-iphadacf963f/ios.

[22] Google, "Google Assistant in Your Car," *Google.com*, 2021. [Online]. Available: https://assistant.google.com/platforms/cars/.

[23] Amazon, "What Is Alexa?" *Amazon.com*, 2021. [Online]. Available: https://developer.amazon.com/en-US/alexa.

[24] H. Mandlekar, S. Purohit, P. Kadam, and H. Tigaiya, "AVA - A Cloud-based Banking Virtual Assistant," *Proceedings of the IEEE International Conference on Intelligent Technologies*, pp. 1–6, Jun. 2021, Hubli, India. DOI: 10.1109/CONIT51480.2021.9498529.

[25] Y. Zhu, Z. C. Gan, and Y. Huang, "The Virtual Sales Assistant in SaaS," *Proceedings of the IEEE International Conference on Advanced Computer Theory and Engineering*, vol. 5, pp. V5-238-V5–241, Aug. 2010, Chengdu, China. DOI: 10.1109/ICACTE.2010.5579806.

[26] A. Kongthon, C. Sangkeettrakarn, S. Kongyoung, and C. Haruechaiyasak, "Implementing an Online Help Desk System Based on Conversational Agent," *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, pp. 450–451, Oct. 2009, Lyon, France. DOI: 10.1145/1643823.1643908.

[27] A. Hutchinson, "Instagram Adds New Text-to-Speech and Voice Effect Options in Reels," *SocialMediaToday*, Nov. 2021. [Online]. Available: https://www.socialmediatoday.com/news/instagram-adds-new-text-to-speech-and-voice-effect-options-in-reels/609925/.

[28] Resemble.AI, "Text to Speech and Custom Voices in TikTok," *Resemble.AI*, Nov. 2021. [Online]. Available: https://www.resemble.ai/tiktok/.

[29] creators, "Attention Reels creators! We know that using sound and audio are one of the funnest aspects of creating a [...] Reel! So today we're launching two new audio tools called Voice Effects and Text to Speech. Swipe through to learn more about how to use them to take your Reels to the next level.," *Instagram*, Nov. 2021. [Online]. Available: https://www.instagram.com/p/CWHL67OPk_H/.

[30] K. Hatton, "How to use TikTok's Text-to-Speech feature," *The Verge*, Jul. 2021. [Online]. Available: https://www.theverge.com/22594929/tiktok-text-to-speech-how-to.

[31] C. Gartenberg, "Disney Partners with TikTok for Official Text-To-Speech Voices from Stitch, Chewbacca, Rocket Raccoon, and More," *The Verge*, Nov. 2021. [Online]. Available: https://www.theverge.com/2021/11/12/22778092/disney-plus-day-tiktok-text-to-speech-voices-characters-marvel.

[32] B. Devore, "'Scream': You Can Now Use the Voice of Ghostface on TikTok's Text-to-Speech," *Collider*, Oct. 2021. [Online]. Available: https://collider.com/scream-ghostface-tiktok-text-to-speech/.

[33] J. Miah, "The Best 9 Effects for Instagram," *Cyberlink*, Nov. 2021. [Online]. Available: https://www.cyberlink.com/blog/video-editing-instagram-tiktok/134/best-instagram-filters-effects.

[34] J. Kim, J. Kong, and J. Son, "Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech," *Proceedings of the International Conference on Machine Learning*, vol. 139, pp. 5530–5540, Jul. 2021, Virtual. [Online]. Available: http://proceedings.mlr.press/v139/kim21f/kim21f.pdf.

[35] T. Wang, R. Fu, J. Yi, *et al.*, "Prosody and Voice Factorization for Few-Shot Speaker Adaptation in the Challenge M2voc 2021," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8603–8607, Jun. 2021, Toronto, Canada. DOI: 10.1109/ICASSP39728.2021.9414427.

[36] V. Popov, I. Vovk, V. Gogoryan, T. Sadekova, and M. Kudinov, "Grad-TTS: A Diffusion Probabilistic Model for Text-to-Speech," *Proceedings of the International Conference on Machine Learning*, vol. 139, pp. 8599–8608, Jul. 2021, Virtual. [Online]. Available: http://proceedings.mlr.press/v139/popov21a/popov21a.pdf.

[37] K. Zhou, B. Sisman, R. Liu, and H. Li, "Seen and Unseen Emotional Style Transfer for Voice Conversion with A New Emotional Speech Dataset," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 920–924, Jun. 2021, Toronto, Canada. DOI: 10.1109/ICASSP39728.2021.9413391.

[38] H. Ajder, G. Patrini, F. Cavalli, and L. Cullen, "The State of Deepfakes: Landscape, Threats, and Impact," *Deeptrace Lab*, pp. 1–27, Sep. 2019. [Online]. Available: https://regmedia.co.uk/2019/10/08/deepfake_report.pdf.

[39] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Niessner, "FaceForensics++: Learning to Detect Manipulated Facial Images," *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1–11, Aug. 2019, Seoul, Korea. DOI: 10.1109/ICCV.2019.00009.

[40] R. Toews, "Deepfakes are Going to Wreck Havoc on Society. We are Not Prepared.," *Forbes*, May 2020. [Online]. Available: https://www.forbes.com/sites/robtoews/2020/05/25/deepfakes-are-going-to-wreak-havoc-on-society-we-are-not-prepared/#67f7615d7494.

[41] B. Allyn, *Deepfake Video of Zelenskyy Could be 'Tip of the Iceberg' in Info War, Experts Warn*, Mar. 2022. [Online]. Available: https://www.npr.org/2022/03/16/1087062648/deepfake-video-zelenskyy-experts-war-manipulation-ukraine-russia.

[42] "Deepfake Video of Volodymyr Zelensky Surrendering Surfaces on Social Media," *The Telegraph*, Mar. 2022. [Online]. Available: https://www.youtube.com/watch?v=X17yrEV5sl4.

[43] B. Smith, "Goldman Sachs, Ozy Media and a $40 Million Conference Call Gone Wrong," *The New York Times*, Sep. 2021. [Online]. Available: https://www.nytimes.com/2021/09/26/business/media/ozy-media-goldman-sachs.html.

[44] L. Yang, D. Holtz, S. Jaffe, *et al.*, "The Effects of Remote Work on Collaboration Among Information Workers," *Nature Human Behavior*, vol. 6, no. 1, pp. 43–54, Sep. 2021. DOI: 10.1038/s41562-021-01196-4.

[45] B. Wang, Y. Liu, J. Qian, and S. K. Parker, "Achieving Effective Remote Working During the COVID-19 Pandemic: A Work Design Perspective," *Applied Psychology*, vol. 70, no. 1, pp. 16–59, Oct. 2020. DOI: 10.1111/apps.12290.

[46] S. Lund, A. Madgavkar, J. Manyika, and S. Smit, "What's Next for Remote Work: An Analysis of 2,000 Tasks, 800 Jobs, and Nine Countries," *McKinsey Global Institute*, Nov. 2020. [Online]. Available: https://www.mckinsey.com/featured-insights/future-of-work/whats-next-for-remote-work-an-analysis-of-2000-tasks-800-jobs-and-nine-countries.

[47] W. Corvey, *Semantic Forensics*, Aug. 2020. [Online]. Available: https://www.darpa.mil/program/semantic-forensics.

[48] S. Stevens, J. Volkmann, and E. Newman, "A Scale for the Measurement of the Psychological Magnitude Pitch," *The Journal of the Acoustical Society of America*, vol. 8, pp. 185–190, 3 Jun. 1937. DOI: 10.1121/1.1915893.

[49] A. Nastasi, "How Long Does it Take to Cross the U.S. on a Horse?" *Hopes & Fears*, 2015. [Online]. Available: http://www.hopesandfears.com/hopes/now/question/215365-how-long-does-it-take-to-cross-the-us-by-horse#:~:text=Under%20normal%20conditions%2C%20the%20trip%20normally%20takes%20about%20four%20months.&text=10%2D12%20gallons%20of%20water,lbs%20require%20for%20good%20health..

[50] Jeffrey, "Top 7: Longest Domestic U.S. Flights," *Weekend Blitz*, May 2013. [Online]. Available: http://weekendblitz.com/top-7-longest-domestic-us-flights/.

[51] J. Jeffrey-Wilensky, "Hypersonic Air Travel Just Took a Step Closer to Reality," *NBC News*, Apr. 2019. [Online]. Available: https://www.nbcnews.com/mach/science/hypersonic-air-travel-just-took-step-closer-reality-ncna993366.

[52] K. Baggaley, "This Hypersonic Airliner Would Take You from Los Angeles to Tokyo in Under Two Hours," *NBC News*, Aug. 2019. [Online]. Available: https://www.nbcnews.com/mach/science/hypersonic-airliner-would-take-you-los-angeles-tokyo-under-two-ncna1045986.

[53] N. Aeronautics and S. Administration, "NASA Fact Sheet: How Scramjets Work," *NASA.gov*, Feb. 2006. [Online]. Available: https://www.nasa.gov/centers/langley/news/factsheets/X43A_2006_5.html#:~:text=A%20scramjet%20(supersonic%2Dcombustion%20ramjet,to%20at%20least%20Mach%2015.

[54] T. Benson, "Speed Regimes: Hypersonic Re-Entry," *National Aeronautics and Space Administration*, May 2021. [Online]. Available: https://www.grc.nasa.gov/www/k-12/BGP/hihyper.html.

[55] S. Gnesin, "The Need for Speed (and Maneuverability)," *Mad Scientist Laboratory*, Jul. 2020. [Online]. Available: https://madsciblog.tradoc.army.mil/257-the-need-for-speed-and-maneuverability/.

[56] Z. Chen, W. Zhang, Z. Xie, X. Xu, and D. Chen, "Recurrent Neural Networks for Automatic Replay Spoofing Attack Detection," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 2052–2056, Apr. 2018, Calgary, Canada. DOI: 10.1109/ICASSP.2018.8462644.

[57] D. Montserrat, H. Hao, S. Yarlagadda, *et al.*, "Deepfakes Detection with Automatic Face Weighting," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 2851–2859, Jun. 2020, Seattle, Washington, USA (Virtual). DOI: 10.1109/CVPRW50498.2020.00342.

[58] D. Güera and E. Delp, "Deepfake Video Detection Using Recurrent Neural Networks," *Proceedings of the IEEE International Conference on Advanced Video and Signal-based Surveillance*, pp. 1–6, Nov. 2018, Auckland, New Zealand. DOI: 10.1109/AVSS.2018.8639163.

[59] E. R. Bartusiak, S. K. Yarlagadda, D. Güera, *et al.*, "Splicing Detection and Localization In Satellite Imagery Using Conditional GANs," *Proceedings of the IEEE Conference on Multimedia Information Processing and Retrieval*, pp. 91–96, Mar. 2019, San Jose, CA, USA. DOI: 10.1109/MIPR.2019.00024.

[60] D. Güera, S. Baireddy, P. Bestagini, S. Tubaro, and E. Delp, "We Need No Pixels: Video Manipulation Detection Using Stream Descriptors," *Proceedings of the International Conference on Machine Learning, Synthetic-Realities: Deep Learning for Detecting AudioVisual Fakes Workshop*, pp. 1–7, Jun. 2019, Long Beach, CA, USA. DOI: 10.48550/arXiv.1906.08743.

[61] A. Chintha, B. Thai, S. J. Sohrawardi, *et al.*, "Recurrent Convolutional Structures for Audio Spoof and Video Deepfake Detection," *The IEEE Journal of Selected Topics in Signal Processing*, pp. 1024–1037, Jun. 2020. DOI: 10.1109/JSTSP.2020.2999185.

[62] B. Bogert, M. Healy, and J. Tukey, "The Quefrency Alanysis of Time Series for Echoes: Cepstrum, Pseudo Autocovariance, Cross-Cepstrum and Saphe Cracking," *Proceedings of the SIAM Symposium on Time Series Analysis*, vol. 15, pp. 209–243, Jun. 1963, New York, NY, USA. [Online]. Available: https://www.researchgate.net/profile/Samuel-Demir-2/post/Anyone-has-this-paper-quefrency-analysis-of-time-series-for-echoes-cepstrum-pseudo-autocovariance-cross-cepstrum-and-saphe-cracking/attachment/5f0493ca4ba4fb0001a4a3c5/AS%3A910684434989062%401594135497855/download/The+quefrency+analysis+of+time+series+for+echoes.pdf.

[63] D. Purves and G. Fitzpatrick, "Neuroscience," *The Audible Spectrum*, vol. 2nd Edition, 2001, Sunderland, MA. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK10924/.

[64] N. Ahmed, T. Natarajan, and K. Rao, "Discrete Cosine Transform," *IEEE Transactions on Computers*, vol. 23, pp. 90–93, Jan. 1974. DOI: 10.1109/T-C.1974.223784.

[65] M. Sahidullah and G. Saha, "Design, Analysis, and Experimental Evaluation of Block based Transformation in MFCC Computation for Speaker Recognition," *Speech Communication*, vol. 54, pp. 543–565, 4 May 2012. DOI: 10.1016/j.specom.2011.11.004.

[66] Z. Chen, Z. Xie, W. Zhang, and X. Xu, "ResNet and Model Fusion for Automatic Spoofing Detection," *Proceedings of the ISCA Interspeech Conference*, pp. 102–106, Aug. 2017, Stockholm, Sweden. DOI: 10.21437/Interspeech.2017-1085.

[67] P. Verma and J. Smith, "Neural Style Transfer for Audio Spectrograms," *Proceedings of the International Conference on Neural Information Processing Systems, Workshop for Machine Learning for Creativity and Design*, pp. 1–3, Dec. 2017, Long Beach, CA, USA. DOI: 10.48550/arXiv.1801.01589.

[68] J. Dennis, H. D. Tran, and H. Li, "Spectrogram Image Feature for Sound Event Classification in Mismatched Conditions," *IEEE Signal Processing Letters*, vol. 18, no. 2, pp. 130–133, Feb. 2011. DOI: 10.1109/LSP.2010.2100380.

[69] Yih Nen Jeng and You-Chi Cheng, "A First Study of Speech Processing via a Novel Mode Decomposition based on the Fourier Sine Spectrum and Spectrogram," *Proceedings of the IEEE Region 10 Conference*, pp. 1–4, Oct. 2007, Taipei, Taiwan. DOI: 10.1109/TENCON.2007.4429153.

[70] S. Greenberg and B. E. D. Kingsbury, "The Modulation Spectrogram: In Pursuit of an Invariant Representation of Speech," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 1647–1650, Apr. 1997, Munich, Germany. DOI: 10.1109/ICASSP.1997.598826.

[71] M. Stolar, M. Lech, R. S. Bolia, and M. Skinner, "Acoustic Characteristics of Emotional Speech Using Spectrogram Image Classification," *Proceedings the IEEE International Conference on Signal Processing and Communication Systems*, pp. 1–5, Dec. 2018, Cairns, Australia. DOI: 10.1109/ICSPCS.2018.8631752.

[72] L. Zheng, Q. Li, H. Ban, and S. Liu, "Speech Emotion Recognition based on Convolution Neural Network Combined with Random Forest," *Proceedings of the Chinese Control And Decision Conference*, pp. 4143–4147, Jun. 2018, Shenyang, China. DOI: 10.1109/CCDC.2018.8407844.

[73] S. Prasomphan, "Detecting Human Emotion via Speech Recognition by using Speech Spectrogram," *Proceedings of the IEEE International Conference on Data Science and Advanced Analytics*, pp. 1–10, Oct. 2015, Paris, France. DOI: 10.1109/DSAA.2015.7344793.

[74] S. Prasomphan, "Improvement of Speech Emotion Recognition with Neural Network Classifier by using Speech Spectrogram," *Proceedings of the International Conference on Systems, Signals and Image Processing*, pp. 73–76, Sep. 2015, London, England. DOI: 10.1109/IWSSIP.2015.7314180.

[75] T. Mittal, U. Bhattacharya, R. Chandra, A. Bera, and D. Manocha, "Emotions Don't Lie: A Deepfake Detection Method using Audio-Visual Affective Cues," *Proceedings of the ACM International Conference on Multimedia*, pp. 2823–2832, Mar. 2020, Seattle, WA, USA. DOI: 10.1145/3394171.3413570.

[76] H. Malik, "Securing Voice-Driven Interfaces Against Fake (Cloned) Audio Attacks," *Proceedings of the IEEE Conference on Multimedia Information Processing and Retrieval*, pp. 512–517, Mar. 2019, San Jose, CA, USA. DOI: 10.1109/MIPR.2019.00104.

[77] L. He, M. Lech, N. C. Maddage, and N. Allen, "Stress Detection Using Speech Spectrograms and Sigma-pi Neuron Units," *Proceedings of the IEEE International Conference on Natural Computation*, pp. 260–264, Aug. 2009, Tianjin, China. DOI: 10.1109/ICNC.2009.59.

[78] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *Proceedings of the International Conference for Learning Representations*, pp. 1–15, May 2015, San Diego, CA, USA. DOI: 10.48550/arXiv.1412.6980.

[79] J. Yamagishi, M. Todisco, M. Sahidullah, *et al.*, "ASVspoof 2019: The 3rd Automatic Speaker Verification Spoofing and Countermeasures Challenge Database," *University of Edinburgh. The Centre for Speech Technology Research*, 2019. DOI: 10.7488/ds/2555.

[80] M. Todisco, J. Yamagishi, M. Sahidullah, *et al.*, "ASVspoof 2019: Automatic Speaker Verification Spoofing and Countermeasures Challenge Evaluation Plan," *Proceedings of the ASVspoof Consortium*, pp. 1–19, Jan. 2019. [Online]. Available: https://www.asvspoof.org/asvspoof2019/asvspoof2019_evaluation_plan.pdf.

[81] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computing*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[82] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative Adversarial Nets," *Proceedings of the International Conference on Neural Information Processing Systems*, vol. 27, pp. 1–9, Dec. 2014, Montréal, Canada. [Online]. Available: https://papers.nips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[83] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," *Proceedings of the International Conference on Learning Representations*, pp. 1–22, May 2021, Virtual. DOI: https://doi.org/10.48550/arXiv.2010.11929.

[84] I. Tolstikhin, N. Houlsby, A. Kolesnikov, *et al.*, "MLP-Mixer: An all-MLP Architecture for Vision," *arXiv:2105.01601*, pp. 1–16, May 2021. DOI: 10.48550/arXiv.2105.01601.

[85] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is All You Need," *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 1–11, Dec. 2017, Long Beach, CA, USA. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[86] A. Hassani, S. Walton, N. Shah, A. Abuduweili, J. Li, and H. Shi, "Escaping the Big Data Paradigm with Compact Transformers," *arXiv:2104.05704*, pp. 1–18, Apr. 2021. DOI: 10.48550/arXiv.2104.05704.

[87] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding," *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4171–4186, Jun. 2019, Minneapolis, MN, USA. DOI: 10.18653/v1/N19-1423.

[88] E. R. Bartusiak and E. J. Delp, "Frequency Domain-Based Detection of Generated Audio," *Proceedings of the IS&T Media Watermarking, Security, and Forensics Conference, Electronic Imaging Symposium*, 273(1)–273(7), Jan. 2021, Burlingame, CA, USA (Virtual). DOI: 10.2352/ISSN.2470-1173.2021.4.MWSF-273.

[89] B. W. Silverman and M. C. Jones, "E. Fix and J.L. Hodges (1951): An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation: Commentary on Fix and Hodges (1951)," *International Statistical Review / Revue Internationale de Statistique*, vol. 57, no. 3, pp. 233–238, Dec. 1989. DOI: 10.2307/1403796.

[90] T. Cover and P. Hart, "Nearest Neighbor Pattern Classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967. DOI: 10.1109/TIT.1967.1053964.

[91] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, pp. 272–297, 1995. DOI: 10.1007/BF00994018.

[92] C. M. Bishop, "Probabilistic Discriminative Models," in *Pattern Recognition and Machine Learning*, Springer-Verlag, New York, USA, 2005, ch. 4.3, pp. 205–210. [Online]. Available: http://users.isr.ist.utl.pt/~wurmd/Livros/school/Bishop%20-%20Pattern%20Recognition%20And%20Machine%20Learning%20-%20Springer%20%202006.pdf.

[93] A. Tharwat, "Classification Assessment Methods," in *Applied Computing and Informatics*, Emerald Publishing Limited, Brighton, United Kingdom, Jul. 2020, pp. 168–192. DOI: 10.1016/j.aci.2018.08.003.

[94] J. Davis and M. Goadrich, "The Relationship between Precision-Recall and ROC Curves," *Proceedings of the International Conference on Machine Learning*, pp. 233–240, Jun. 2006, Pittsburgh, Pennsylvania, USA. DOI: 10.1145/1143844.1143874.

[95] K. Koutini, J. Schlüter, H. Eghbal-zadeh, and G. Widmer, "Efficient Training of Audio Transformers with Patchout," *Proceedings of the ISCA Interspeech Conference*, pp. 1–5, Sep. 2022, Incheon, Korea. [Online]. Available: https://www.isca-speech.org/archive/pdfs/interspeech_2022/koutini22_interspeech.pdf.

[96] Y. Gong, C.-I. J. Lai, Y.-A. Chung, and J. Glass, "SSAST: Self-Supervised Audio Spectrogram Transformer," *arXiv:2110.09784*, pp. 1–11, Oct. 2021. DOI: 10.48550/arXiv.2110.09784.

[97] G. Hua, A. B. J. Teoh, and H. Zhang, "Towards End-to-End Synthetic Speech Detection," *IEEE Signal Processing Letters*, vol. 28, pp. 1265–1269, Jun. 2021. DOI: 10.1109/LSP.2021.3089437.

[98] H. Hao, E. R. Bartusiak, D. Güera, *et al.*, "Deepfake Detection Using Multiple Data Modalities," in *Handbook of Digital Face Manipulation and Detection - From DeepFakes to Morphing Attacks, Series on Advances in Computer Vision and Pattern Recognition*, Springer, Cham, Jan. 2022, pp. 235–254. DOI: 10.1007/978-3-030-87664-7_11.

[99] E. R. Bartusiak and E. J. Delp, "Synthesized Speech Detection Using Convolutional Transformer-Based Spectrogram Analysis," *Proceedings of the IEEE Asilomar Conference on Signals, Systems, and Computers*, pp. 1426–1430, Oct. 2021, Asilomar, CA, USA. DOI: 10.1109/IEEECONF53345.2021.9723142.

[100] E. Conti, D. Salvi, C. Borrelli, *et al.*, "Deepfake Speech Detection Through Emotion Recognition: A Semantic Approach," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8962–8966, May 2022, Singapore. DOI: 10.1109/ICASSP43922.2022.9747186.

[101] Y. Gong, Y.-A. Chung, and J. Glass, "AST: Audio Spectrogram Transformer," *Proceedings of the ISCA Interspeech Conference*, pp. 571–575, Aug. 2021, Brno, Czech Republic. DOI: 10.21437/Interspeech.2021-698.

[102] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," *Proceedings of the International Conference on Learning Representations*, pp. 1–19, May 2019, New Orleans, LA, USA. DOI: 10.48550/arXiv.1711.05101.

[103] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, *et al.*, "Audio Set: An Ontology and Human-Labeled Dataset for Audio Events," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 776–780, Mar. 2017, New Orleans, LA. DOI: 10.1109/ICASSP.2017.7952261.

[104] podcast.ai, *Joe Rogan Interviews Steve Jobs*, https://share.transistor.fm/s/22f16c7f.

[105] L. Rabiner and R. Schafer, *Theory and Applications of Digital Speech Processing*, 1st. USA: Prentice Hall Press, 2010.

[106] L. van der Maaten and G. E. Hinton, "Visualizing High-Dimensional Data Using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, Nov. 2008. [Online]. Available: https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf.

[107] K. Bhagtani, A. K. S. Yadav, E. R. Bartusiak, *et al.*, "An Overview of Recent Work in Media Forensics: Methods and Threats," *arXiv:2204.12067*, pp. 1–17, Apr. 2022. DOI: 10.48550/arXiv.2204.12067.

[108] K. Bhagtani, A. K. S. Yadav, E. R. Bartusiak, *et al.*, "An Overview of Recent Work in Multimedia Forensics," *Proceedings of the IEEE Conference on Multimedia Information Processing and Retrieval*, pp. 324–329, Aug. 2022, Virtual. DOI: 10.1109/MIPR54900.2022.00064.

[109] R. Buchholz, C. Kraetzer, and J. Dittmann, "Microphone Classification Using Fourier Coefficients," *International Workshop on Information Hiding*, pp. 235–246, Jun. 2009, Darmstadt, Germany. DOI: 10.1007/978-3-642-04431-1_17.

[110] D. Luo, P. Korus, and J. Huang, "Band Energy Difference for Source Attribution in Audio Forensics," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 9, pp. 2179–2189, Sep. 2018. DOI: 10.1109/TIFS.2018.2812185.

[111] C. Borrelli, P. Bestagini, F. Antonacci, A. Sarti, and S. Tubaro, "Synthetic Speech Detection through Short-Term and Long-Term Prediction Traces," *EURASIP Journal on Information Security*, vol. 2021, no. 2, pp. 1–14, Apr. 2021. DOI: 10.1186/s13635-021-00116-3.

[112] Z. Leng, M. Tan, C. Liu, *et al.*, "PolyLoss: A Polynomial Expansion Perspective of Classification Loss Functions," *Proceedings of the International Conference on Learning Representations*, pp. 1–16, Apr. 2022, Virtual. DOI: 10.48550/arXiv.2204.12511.

[113] IEEE, *IEEE Signal Processing Cup 2022*, 2022. [Online]. Available: https://signalprocessingsociety.org/community-involvement/signal-processing-cup.

[114] Nvidia, *Riva*. [Online]. Available: https://developer.nvidia.com/riva.

[115] A. Łańcucki, "FastPitch: Parallel Text-to-Speech with Pitch Prediction," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6588–6592, Jun. 2021, Toronto, Canada. DOI: 10.1109/ICASSP39728.2021.9413889.

[116] Y. Ren, C. Hu, X. Tan, *et al.*, "FastSpeech 2: Fast and High-Quality End-to-End Text to Speech," *Proceedings of the International Conference on Learning Representations*, pp. 1–15, May 2021, Virtual. DOI: 10.48550/arXiv.2006.04558.

[117] J. Kim, S. Kim, J. Kong, and S. Yoon, "Glow-TTS: A Generative Flow for Text-to-Speech via Monotonic Alignment Search," *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 1–14, Dec. 2020, Virtual. DOI: 10.48550/arXiv.2005.11129.

[118] *Google Cloud Text-to-Speech*. [Online]. Available: https://cloud.google.com/text-to-speech.

[119] Y. Wang, R. J. Skerry-Ryan, D. Stanton, *et al.*, "Tacotron: Towards End-to-End Speech Synthesis," *Proceedings of the ISCA Interspeech Conference*, pp. 4006–4010, Aug. 2017, Stockholm, Sweden. DOI: 10.21437/Interspeech.2017-1452.

[120] J. Shen, R. Pang, R. J. Weiss, *et al.*, "Natural TTS Synthesis by Conditioning Wavenet on Mel Spectrogram Predictions," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4779–4783, Apr. 2018, Calgary, Canada. DOI: 10.1109/ICASSP.2018.8461368.

[121] S. Beliaev, Y. Rebryk, and B. Ginsburg, "TalkNet: Fully-Convolutional Non-Autoregressive Speech Synthesis Model," *arXiv:2005.05514*, pp. 1–5, May 2020. DOI: 10.48550/arXiv.2005.05514.

[122] O. Tatanov, S. Beliaev, and B. Ginsburg, "Mixer-TTS: Non-Autoregressive, Fast and Compact Text-to-Speech Model Conditioned on Language Model Embeddings," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 7482–7486, May 2022, Singapore. DOI: 10.1109/ICASSP43922.2022.9746107.

[123] J. Vainer and O. Dušek, "SpeedySpeech: Efficient Neural Speech Synthesis," *Proceedings of the ISCA Interspeech Conference*, pp. 1–5, Oct. 2020, Shanghai, China. DOI: 10.48550/arXiv.2008.03802.

[124] E. Hoogeboom, R. van den Berg, and M. Welling, "Emerging Convolutions for Generative Normalizing Flows," *Proceedings of the International Conference on Machine Learning*, pp. 1–10, Jun. 2019, Long Beach, CA, USA. [Online]. Available: http://proceedings.mlr.press/v97/hoogeboom19a/hoogeboom19a.pdf.

[125] D. P. Kingma and P. Dhariwal, "Glow: Generative Flow with Invertible 1x1 Convolutions," *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 10 236–10 245, Dec. 2018, Montréal, Canada. [Online]. Available: https://papers.nips.cc/paper/2018/file/d139db6a236200b21cc7f752979132d0-Paper.pdf.

[126] A. van den Oord, S. Dieleman, H. Zen, *et al.*, "WaveNet: A Generative Model for Raw Audio," *arXiv:1609.03499*, pp. 1–15, Sep. 2016. DOI: 10.48550/arXiv.1609.03499.

[127] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 1–9, Dec. 2014, Montréal, Canada. [Online]. Available: https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf.

[128] K. Kumar, R. Kumar, T. de Boissiere, *et al.*, "MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis," *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 14 910–14 921, Dec. 2019, Vancouver, Canada. [Online]. Available: https://papers.nips.cc/paper/2019/file/6804c9bca0a615bdb9374d00a9fcba59-Paper.pdf.

[129] J. Kong, J. Kim, and J. Bae, "HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis," *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 1–12, Dec. 2020, Virtual. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/c5d736809766d46260d816d8dbc9eb44-Paper.pdf.

[130] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *Proceedings of the International Conference on Learning Representations*, pp. 1–14, Apr. 2014, Banff, Canada. DOI: 10.48550/arXiv.1312.6114.

[131] K. Ito and L. Johnson, *The LJ Speech Dataset*, https://keithito.com/LJ-Speech-Dataset/, 2017.

[132] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "LibriSpeech: An ASR Corpus Based on Public Domain Audio Books," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5206–5210, Apr. 2015, South Brisbane, Australia. DOI: 10.1109/ICASSP.2015.7178964.

[133] J. Yamagishi, *English Multi-Speaker Corpus for CSTR Voice Cloning Toolkit*, 2012. [Online]. Available: https://datashare.ed.ac.uk/handle/10283/3443.

[134] Missile Defense Advocacy Alliance, *Hypersonic Weapon Basics*, Apr. 2020. [Online]. Available: https://missiledefenseadvocacy.org/missile-threat-and-proliferation/missile-basics/hypersonic-missiles/.

[135] R. H. Speier, G. Nacouzi, C. A. Lee, and R. M. Moore, *Hypersonic Missile Nonproliferation: Hindering the Spread of a New Class of Weapons*, 2017. [Online]. Available: https://www.rand.org/content/dam/rand/pubs/research_reports/RR2100/RR2137/RAND_RR2137.pdf.

[136] H.-L. Besser, D. Göge, M. Huggins, A. Shaffer, and D. Zimper, "Hypersonic Vehicles: Game Changers for Future Warfare?" *Transforming Joint Air Power: The Journal of the Joint Air Power Competence Centre*, vol. 24, pp. 11–27, 2017. [Online]. Available: https://www.japcc.org/wp-content/uploads/JAPCC_J24_screen.pdf.

[137] A. F. Woolf, "Conventional Prompt Global Strike and Long-Range Ballistic Missiles: Background and Issues," *U.S. Congressional Research Service*, vol. R41464, Feb. 2020. [Online]. Available: https://crsreports.congress.gov/product/pdf/R/R41464.

[138] United States Office of the Secretary of Defense, *2019 Missile Defense Review*, United States Department of Defense, Apr. 2020. [Online]. Available: https://media.defense.gov/2019/Jan/17/2002080666/-1/-1/1/2019-MISSILE-DEFENSE-REVIEW.pdf.

[139] J. Sola and J. Sevilla, "Importance of Input Data Normalization for the Application of Neural Networks to Complex Industrial Problems," *IEEE Transactions on Nuclear Science*, vol. 44, no. 3, pp. 1464–1468, 1997. DOI: 10.1109/23.589532.

[140] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, USA, 2001, pp. 1–764. [Online]. Available: https://web.stanford.edu/~hastie/ElemStatLearn//printings/ESLII_print12.pdf.

[141] S. M. Piryonesi and T. E. El-Diraby, "Role of Data Analytics in Infrastructure Asset Management: Overcoming Data Size and Quality Problems," *Journal of Transportation Engineering, Part B: Pavements*, vol. 146, no. 2, pp. 1–17, Jun. 2020. DOI: 10.1061/JPEODX.0000175.

[142] R. Arian, A. Hariri, A. Mehridehnavi, A. Fassihi, and F. Ghasemi, "Protein Kinase Inhibitors' Classification using K-Nearest Neighbor Algorithm," *Computational Biology and Chemistry*, vol. 86, pp. 1–7, Jun. 2020. DOI: 10.1016/j.compbiolchem.2020.107269.

[143] A. R. Persico, C. V. Ilioudis, C. Clemente, and J. J. Soraghan, "Novel Classification Algorithm for Ballistic Target Based on HRRP Frame," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, no. 6, pp. 3168–3189, Mar. 2019. DOI: 10.1109/TAES.2019.2905281.

[144] A. R. Persico, C. Ilioudis, C. Clemente, and J. Soraghan, "Novel Approach for Ballistic Targets Classification from HRRP Frame," *Proceedings of the IEEE Sensor Signal Processing for Defence Conference*, pp. 1–5, Dec. 2017, London, England. DOI: 10.1109/SSPD.2017.8233248.

[145] A. R. Persico, C. Clemente, L. Pallotta, A. De Maio, and J. Soraghan, "Micro-Doppler Classification of Ballistic Threats using Krawtchouk Moments," *Proceedings of the IEEE Radar Conference*, pp. 1–6, May 2016, Philadelphia, PA, USA. DOI: 10.1109/RADAR.2016.7485086.

[146] U. K. Singh, V. Padmanabhan, and A. Agarwal, "Dynamic Classification of Ballistic Missiles using Neural Networks and Hidden Markov Models," *Applied Soft Computing*, vol. 19, pp. 280–289, Jun. 2014. DOI: 10.1016/j.asoc.2014.02.015.

[147] U. K. Singh, V. Padmanabhan, and A. Agarwal, "A Novel Method for Training and Classification of Ballistic and Quasi-Ballistic Missiles in Real-Time," *Proceedings of the IEEE International Joint Conference on Neural Networks*, pp. 1–8, Aug. 2013, Dallas, TX, USA. DOI: 10.1109/IJCNN.2013.6707115.

[148] U. K. Singh and V. Padmanabhan, "Training and Classification of Ballistic Missiles using Hidden Markov Model," *Proceedings of the IEEE International Conference on Contemporary Computing*, pp. 301–306, Aug. 2013, Noida, India. DOI: 10.1109/IC3.2013.6612209.

[149] S. Park, J. Jeong, C. Ryoo, and K. Choi, "Detection and Classification of a Ballistic Missile in Ascent Phase," *Proceedings of the IEEE International Conference on Control, Automation and Systems*, pp. 1235–1238, Oct. 2011, Gyeonggi-do, South Korea. [Online]. Available: https://ieeexplore.ieee.org/document/6106116.

[150] A. Farina, L. Timmoneri, and D. Vigilante, "Classification and Launch-Impact Point Prediction of Ballistic Target via Multiple Model Maximum Likelihood Estimator (MM-MLE)," *Proceedings of the IEEE Conference on Radar*, pp. 1–5, Apr. 2006, Verona, NY, USA. DOI: 10.1109/RADAR.2006.1631895.

[151] V. Jithesh, M. J. Sagayaraj, and K. G. Srinivasa, "LSTM Recurrent Neural Networks for High Resolution Range Profile Based Radar Target Classification," *Proceedings of the International Conference on Computational Intelligence Communication Technology*, pp. 1–6, Feb. 2017, Ghaziabad, India. DOI: 10.1109/CIACT.2017.7977298.

[152] N. E. Gaiduchenko and P. A. Gritsyk, "Hypersonic Vehicle Trajectory Classification Using Convolutional Neural Network," *Proceedings of the IEEE International Conference on Engineering and Telecommunication*, pp. 1–4, Nov. 2019, Dolgoprudny, Russia. DOI: 10.1109/EnT47717.2019.9030537.

[153] K. Fukushima, "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position," *Biological Cybernetics*, vol. 36, pp. 193–202, 1980. DOI: 10.1007/BF00344251.

[154] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016, ch. Chapter 9: Convolutional Networks, pp. 326–366. [Online]. Available: http://www.deeplearningbook.org.

[155] M. Ben-Bassat, K. L. Klove, and M. H. Weil, "Sensitivity Analysis in Bayesian Classification Models: Multiplicative Deviations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-2, no. 3, pp. 261–266, May 1980. DOI: 10.1109/TPAMI.1980.4767015.

[156] P. Domingos and M. Pazzani, "On the Optimality of the Simple Bayesian Classifier under Zero-One Loss," *Machine Learning*, vol. 29, pp. 103–130, 1997. DOI: 10.1023/A:1007413511361.

[157] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016, ch. Chapter 7: Regularization for Deep Learning, pp. 224–270. [Online]. Available: http://www.deeplearningbook.org.

[158] H. Noh, T. You, J. Mun, and B. Han, "Regularizing Deep Neural Networks by Noise: Its Interpretation and Optimization," *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 5115–5124, Dec. 2017, Long Beach, CA, USA. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/217e342fc01668b10cb1188d40d3370e-Paper.pdf.

[159] K. Audhkhasi, O. Osoba, and B. Kosko, "Noise-Enhanced Convolutional Neural Networks," *Neural Networks*, vol. 78, pp. 15–23, Jun. 2016, Special Issue on "Neural Network Learning in Big Data". DOI: 10.1016/j.neunet.2015.09.014.

[160] W. Chen, H. Zhou, W. Yu, and L. Yang, *Steady Glide Dynamics and Guidance of Hypersonic Vehicle*. Science Press, Beijing, 2021, pp. 1–482. DOI: 10.1007/978-981-15-8901-0.

[161] E. Gillard, "Hypersonic Flight's Challenges, Successes and Opportunities Discussed at Colloquium," *National Aeronautics and Space Administration*, Sep. 2020. [Online]. Available: https://www.nasa.gov/feature/langley/hypersonic-flight-s-challenges-successes-and-opportunities-discussed-at-colloquium.

[162] T. Benson, "Speed Regimes: Low Hypersonic," *National Aeronautics and Space Administration*, May 2021. [Online]. Available: https://www.grc.nasa.gov/www/k-12/BGP/lowhyper.html.

[163] R. W. Orloff, "Apollo by the Numbers: A Statistical Reference," *NASA History Division*, Oct. 2000, NASA SP-2000-4029. [Online]. Available: https://history.nasa.gov/SP-4029.pdf.

[164] A. Davies, "Elon Musk's Rocket Travel Plan Is Def Possible, Def Bananas," *Wired*, Sep. 2017. [Online]. Available: https://www.wired.com/story/elon-musk-spacex-rocket-travel-plan/.

[165] E. Musk, "∼1000, as all seats would be "coach" & no toilets, pilot area or food galley needed. Most flights would only be 15 to 20 mins. It's basically an ICBM traveling at Mach 25 that lands.," *Twitter*, Jun. 2019. [Online]. Available: https://twitter.com/elonmusk/status/1144004310503530496.

[166] SpaceX, "Earth to Earth Transportation," *SpaceX.com*, Aug. 2021. [Online]. Available: https://www.spacex.com/human-spaceflight/earth/index.html.

[167] M. Bunn, "Review of U.S. Military Research and Development, 1984," *American Political Science Review*, vol. 79, no. 4, pp. 67–116, Dec. 1985. DOI: 10.1017/S0003055400239002.

[168] F. Graziani, S. Trofimov, and S. Battistini, "Applied Astrodynamics," in *Cubesat Handbook: From Mission Design to Operations*, Academic Press, Cambridge, MA, USA, Jan. 2021, pp. 35–52. DOI: 10.1016/B978-0-12-817884-3.00001-1.

[169] J. L. Russell, "Kepler's Laws of Planetary Motion: 1609-1666," *The British Journal for the History of Science*, vol. 2, pp. 1–24, Jun. 1964. [Online]. Available: http://www.jstor.org/stable/4025081.

[170] A. Benavoli, L. Chisci, and A. Farina, "Tracking of a Ballistic Missile with A-Priori Information," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 3, pp. 1000–1016, Jul. 2007. DOI: 10.1109/TAES.2007.4383589.

[171] R. Ji, Y. Liang, L. Xu, and Z. Wei, "Trajectory Prediction of Ballistic Missiles Using Gaussian Process Error Model," *Chinese Journal of Aeronautics*, pp. 458–469, May 2021. DOI: 10.1016/j.cja.2021.05.011.

[172] J. A. Isaacson and D. R. Vaughan, *Estimation and Prediction of Ballistic Missile Trajectories*. RAND Corporation, Santa Monica, CA, USA, 1996, pp. 1–98. [Online]. Available: https://www.rand.org/content/dam/rand/pubs/monograph_reports/2006/MR737.pdf.

[173] F. J. Regan and S. M. Anandakrishnan, *Dynamics of Atmospheric Re-Entry*. American Institute of Aeronautics and Astronautics, Jan. 1993, pp. 1–591. DOI: 10.2514/5.9781600861741.0000.0000.

[174] S. Ayhan and H. Samet, "Aircraft Trajectory Prediction Made Easy with Predictive Analytics," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 21–30, Aug. 2016, San Francisco, CA, USA. DOI: 10.1145/2939672.2939694.

[175] D. Lee, M. Tahk, C. Lee, and Y. Kim, "Singularity-Free Analytic Solution of Ballistic Trajectory with Quadratic Drag," *Proceedings of the IEEE Mediterranean Conference on Control and Automation*, pp. 249–253, Sep. 2020, Saint-Raphaël, France. DOI: 10.1109/MED48518.2020.9182863.

[176] G. Linyu, W. Nan, M. Fankun, and H. Jiaying, "A Analytical Method of Trajectory Prediction Considering J2 Perturbations and Including Short-Period Terms," *Proceedings of the IEEE International Conference on Control, Automation and Robotics*, pp. 498–503, Apr. 2018, Auckland, New Zealand. DOI: 10.1109/ICCAR.2018.8384727.

[177] D. G. Lee, K. S. Cho, and J. H. Shin, "A Simple Prediction Method of Ballistic Missile Trajectory to Designate Search Direction and its Verification using a Test-bench," *Proceedings of the IEEE Asian Control Conference*, pp. 1–7, May 2015, Kota Kinabalu, Malaysia. DOI: 10.1109/ASCC.2015.7244461.

[178] Q. Lei, L. Junlong, and Z. Di, "Tracking Filter and Prediction for Non-Ballistic Target HTV-2 in Near Space," *Proceedings of the Chinese Control and Decision Conference*, pp. 3556–3561, May 2015, Qingdao, China. DOI: 10.1109/CCDC.2015.7162539.

[179] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, Mar. 1960. DOI: 10.1115/1.3662552.

[180] K. R. Moon, T. H. Kim, and T.-L. Song, "Comparison of Ballistic-Coefficient-Based Estimation Algorithms for Precise Tracking of a Re-Entry Vehicle and its Impact Point Prediction," *Journal of Astronomy and Space Sciences*, vol. 29, no. 4, pp. 363–374, Dec. 2012. DOI: 10.5140/JASS.2012.29.4.363.

[181] C. Han and J. Xiong, "Method of Trajectory Prediction for Unpowered Gliding Hypersonic Vehicle in Gliding Phase," *Proceedings of the IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference*, pp. 262–266, Oct. 2016, Xi'an, China. DOI: 10.1109/IMCEC.2016.7867213.

[182] Y. Luo, X. Tan, H. Wang, Z. Qu, and T. Li, "Trajectory Prediction of Hypersonic Vehicles based on Control Quantity Prediction," *Proceedings of the IEEE Information Technology, Networking, Electronic and Automation Control Conference*, vol. 1, pp. 87–91, Jun. 2020, Chongqing, China. DOI: 10.1109/ITNEC48623.2020.9084956.

[183] Y. Xie, X. Zhuang, Z. Xi, and H. Chen, "Dual-Channel and Bidirectional Neural Network for Hypersonic Glide Vehicle Trajectory Prediction," *IEEE Access*, vol. 9, pp. 92 913–92 924, Jun. 2021. DOI: 10.1109/ACCESS.2021.3092515.

[184] N. Chomsky, "Three Models for the Description of Language," *IEEE Transactions on Information Theory*, vol. 2, no. 3, pp. 113–124, Sep. 1956. DOI: 10.1109/TIT.1956.1056813.

[185] N. Chomsky, *Syntatic Structures*. Mouton & Co. Publishers, Den Haag, Netherlands, Feb. 1957.

[186] N. Chomsky, "On Certain Formal Properties of Grammars," *Information and Control*, vol. 2, no. 2, pp. 137–167, Jun. 1959. DOI: 10.1016/S0019-9958(59)90362-6.

[187] Y. Sakakibara, M. Brown, R. Hughey, *et al.*, "Stochastic Context-Free Grammars for tRNA Modeling," *Nucleic Acids Research*, vol. 22, no. 23, pp. 5112–5120, Nov. 1994. DOI: 10.1093/nar/22.23.5112.

[188] L. Grate, "Automatic RNA Secondary Structure Determination with Stochastic Context-Free Grammars," *Proceedings of the AAAI International Conference on Intelligent Systems for Molecular Biology*, vol. 3, pp. 136–144, Jul. 1995, Cambridge, United Kingdom.

[189] F. Lefebvre, "An Optimized Parsing Algorithm Well Suited to RNA Folding," *Proceedings of the AAAI International Conference on Intelligent Systems for Molecular Biology*, pp. 222–230, Jul. 1995, Cambridge, United Kingdom. [Online]. Available: http://www.aaai.org/Library/ISMB/1995/ismb95-027.php.

[190] F. Lefebvre, "A Grammar-Based Unification of Several Alignment and Folding Algorithms," *Proceedings of the AAAI International Conference on Intelligent Systems for Molecular Biology*, pp. 143–154, Jun. 1996, St. Louis, MO, USA.

[191] S. Qi, S. Huang, P. Wei, and S.-C. Zhu, "Predicting Human Activities Using Stochastic Grammar," pp. 1173–1181, Nov. 2017, Venice, Italy. DOI: 10.1109/ICCV.2017.132.

[192] S. Qi, B. Jia, and S.-C. Zhu, "Generalized Earley Parser: Bridging Symbolic Grammars and Sequence Data for Future Prediction," *Proceedings of the International Conference on Machine Learning*, pp. 4171–4179, Jul. 2018, Stockholm, Sweden. [Online]. Available: http://proceedings.mlr.press/v80/qi18a/qi18a.pdf.

[193] S. Qi, B. Jia, S. Huang, P. Wei, and S.-C. Zhu, "A Generalized Earley Parser for Human Activity Parsing and Prediction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 8, pp. 2538–2554, Feb. 2020. DOI: 10.1109/TPAMI.2020.2976971.

[194] B. Rothrock, S. Park, and S.-C. Zhu, "Integrating Grammar and Segmentation for Human Pose Estimation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3214–3221, Jun. 2013, Portland, OR, USA. DOI: 10.1109/CVPR.2013.413.

[195] M. Fanaswala and V. Krishnamurthy, "Detection of Anomalous Trajectory Patterns in Target Tracking via Stochastic Context-Free Grammars and Reciprocal Process Models," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 1, pp. 76–90, Feb. 2013. DOI: 10.1109/JSTSP.2012.2233459.

[196] G. Astolfi, F. P. C. Rezende, J. V. D. A. Porto, E. T. Matsubara, and H. Pistori, "Syntactic Pattern Recognition in Computer Vision: A Systematic Review," *ACM Computing Surveys*, vol. 54, no. 3, pp. 1–35, Apr. 2022. DOI: 10.1145/3447241.

[197] N. R. Brisaboa, A. Gómez-Brandón, G. Navarro, and J. R. Paramá, "GraCT: A Grammar-based Compressed Index for Trajectory Data," *Information Sciences*, vol. 483, pp. 106–135, May 2019. DOI: 10.1016/j.ins.2019.01.035.

[198] J. López-Leonés, M. A. Vilaplana, E. Gallo, F. A. Navarro, and C. Querejeta, "The Aircraft Intent Description Language: A Key Enabler for Air-Ground Synchronization in Trajectory-Based Operations," *Proceedings of IEEE/AIAA Digital Avionics Systems Conference*, pp. 1.D.4-1-1.D.4–12, Oct. 2007, Dallas, TX, USA. DOI: 10.1109/DASC.2007.4391836.

[199] Y. Sakakibara, "Probabilistic Context-Free Grammars," in *Encyclopedia of Machine Learning*. Springer, Boston, MA, USA, 2011, pp. 802–805. DOI: 10.1007/978-0-387-30164-8_669.

[200] I. Chiswell and W. Hodges, *Mathematical Logic*. Oxford University Press, New York, USA, Jul. 2007, pp. 1–259. [Online]. Available: http://perso.ens-lyon.fr/jacques.jayez/Cours/Proving/Mathematical_Logic_(Ian_Chiswell,_Wilfrid_Hodges).pdf.

[201] Y. Fan, W. Zhu, and G. Bai, "A Cost-Effective Tracking Algorithm for Hypersonic Glide Vehicle Maneuver Based on Modified Aerodynamic Model," *Applied Sciences*, vol. 6, pp. 1–17, 10 Oct. 2016. DOI: 10.3390/app6100312.

[202] G. Richie, "The Common Aero Vehicle - Space Delivery System of the Future," *Proceedings of the AIAA Space Technology Conference and Exposition*, pp. 28–30, Sep. 1999, Albuquerque, NM, USA. DOI: 10.2514/6.1999-4435.

[203] T. H. Phillips, "A Common Aero Vehicle (CAV): Model, Description, and Employment Guide," *Schafer Corporation for AFRL and AFSPC*, Jan. 2003.

[204] Z. Wang, X. X. Cheng, and H. Li, "Hypersonic Skipping Trajectory Planning for High L/D Gliding Vehicles," *Proceedings of the AIAA International Space Planes and Hypersonics Technologies Conference*, Mar. 2017, Xiamen, China. DOI: 10.2514/6.2017-2135.

[205] Y. Fan, W. Zhu, and G. Bai, "Aerodynamic Coefficients Models of Hypersonic Vehicle Based on Aero Database," *Proceedings of the IEEE International Conference on Pervasive Computing, Signal Processing, and Applications*, vol. 6, pp. 1001–1004, 10 Nov. 2010, Harbin, China. DOI: 10.1109/PCSPA.2010.247.

[206] National Imagery and Mapping Agency, "Department of Defense World Geodetic System 1984: Its Definition, and Relationship with Local Geodetic Systems," *Department of Defense*, pp. 1–169, Jan. 2000, Washington DC, USA. [Online]. Available: https://apps.dtic.mil/sti/pdfs/ADA280358.pdf.

[207] Z. Solan, D. Horn, E. Ruppin, and S. Edelman, "Unsupervised Learning of Natural Languages," *Proceedings of the National Academy of Sciences*, vol. 102, no. 33, pp. 11 629–11 634, Aug. 2005. DOI: 10.1073/pnas.0409746102.

[208] E. R. Bartusiak, N. X. Nguyen, M. W. Chan, M. L. Comer, and E. J. Delp, "A Machine Learning Approach to Classify Hypersonic Vehicle Trajectories," *Proceedings of the IEEE Aerospace Conference*, pp. 1–14, Mar. 2021, Big Sky, MT, USA (Virtual). DOI: 10.1109/AERO50100.2021.9438274.

[209] J. Earley, "An Efficient Context-Free Parsing Algorithm," *Communications of the ACM*, vol. 13, no. 2, pp. 94–102, Feb. 1970. DOI: 10.1145/362007.362035.

[210] S. Baireddy, S. R. Desai, J. L. Mathieson, *et al.*, "Spacecraft Time-Series Anomaly Detection Using Transfer Learning," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Workshop on AI for Space*, pp. 1951–1960, Jun. 2021, Nashville, TN, USA. DOI: 10.1109/CVPRW53098.2021.00223.

[211] T. Li, M. L. Comer, E. J. Delp, *et al.*, "Anomaly Scoring for Prediction-Based Anomaly Detection in Time Series," *Proceedings of the IEEE Aerospace Conference*, pp. 1–7, Mar. 2020, Big Sky, MT, USA. DOI: 10.1109/AERO47225.2020.9172442.

[212] K. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, Cambridge, MA, USA, 2012, pp. 1–1098. [Online]. Available: http://noiselab.ucsd.edu/ECE228/Murphy_Machine_Learning.pdf.

[213] Missile Defense Agency, *GPI Scenario Animation*, Jun. 2021. [Online]. Available: https://www.youtube.com/watch?v=-q-ieXZgrhY.

[214] C. Manning and H. Schutze, *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA, May 1999.

[215] E. R. Bartusiak, H. Hao, M. A. Jacobs, *et al.*, "A Stochastic Grammar Approach to Predict Flight Phases of a Hypersonic Glide Vehicle," *Proceedings of the IEEE Aerospace Conference*, pp. 1–15, Mar. 2022, Big Sky, MT, USA. DOI: 10.1109/AERO53065.2022.9843362.

[216] A. Gersho and R. M. Gray, "Vector Quantization and Signal Compression," in *The Springer International Series in Engineering and Computer Science*. Springer New York, NY, USA, 1992, vol. 159, pp. 1–732. DOI: 10.1007/978-1-4615-3626-0.

[217] T. Antony, M. Grant, M. Sparapany, *et al.*, *Beluga*, 2021. [Online]. Available: https://github.com/Rapid-Design-of-Systems-Laboratory/beluga.

[218] T. Antony, "Large Scale Constrained Trajectory Optimization Using Indirect Methods," *Purdue University*, pp. 1–202, May 2018, West Lafayette, IN, USA. [Online]. Available: https://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=2923&context=open_access_dissertations.

[219] K. Mall, M. J. Grant, and E. Taheri, "Uniform Trigonometrization Method for Optimal Control Problems with Control and State Constraints," *Journal of Spacecraft and Rockets*, vol. 57, pp. 995–1007, 5 Jul. 2020. DOI: 10.2514/1.A34624.

[220] N. X. Vinh, A. Busemann, and R. D. Culp, "Hypersonic and Planetary Entry Flight Mechanics," in *NASA Scientific and Technical Information/Recon Repository*. University of Michigan Press, 1980, vol. A, pp. 1–357.

VITA

Emily R. Bartusiak is a Ph.D. Student in Electrical Engineering at Purdue University. She is also a proud, two-time graduate of Purdue with both a Master's degree in Electrical Engineering and a Bachelor's degree in Electrical Engineering with a minor in Management. During her Master's degree, Emily focused on machine learning and image processing concepts, which culminated in her thesis, titled *An Adversarial Approach to Spliced Forgery Detection and Localization in Satellite Imagery*. In this work, she employed a deep learning approach to identify forgeries in satellite images. As a Ph.D. student, Emily investigates machine learning and deep learning for media forensics, aerospace, and biomedical research.

Emily also excels in industry. She conducted eight (8) internships: Stanford University in Palo Alto, CA (2021); Apple (in the AI/ML organization) in Seattle, WA (2020); Lockheed Martin Space (in the Advanced Technology Center) in Palo Alto, CA (2019); Qualcomm in San Diego, CA (2018, 2017, 2016); and Motorola Solutions in Schaumburg, IL (2015, 2014). The internships at Stanford, Apple, and Lockheed Martin were AI/ML research internships. At Lockheed Martin, Emily was deemed an Intern Presentation Winner for her clear communication of complex concepts and engaging style as she presented work conducted during the internship. Most recently, she was named a NCWIT Collegiate Award Finalist in January 2022 for her technical contributions and research.

Service is important to Emily. She currently serves on the Executive Board of the Electrical and Computer Engineering Graduate Association (ECEGSA) as Vice President and as a board member for the Woodlands Academy Alumnae Association (WAAA). She previously served on the Executive Boards of the Purdue Chapter of IEEE-Eta Kappa Nu (IEEE-HKN) Electrical and Computer Engineering Honor Society and the Purdue Chapter of Tau Beta Pi (TBP) Engineering Honor Society. Emily also served as a mentor in the Purdue University GradTrack program to help prepare underrepresented undergraduates for graduate school. She received two awards from IEEE-HKN for her unparalleled initiative and dedication to the chapter. Furthermore, the College of Engineering at Purdue University honored her with the Magoon Award for excellence in teaching for her time as a Teaching Assistant for a Purdue program known as Vertically Integrated Projects (VIP).

PUBLICATIONS

Journal Publications Resulting from this Dissertation:

1. **E. R. Bartusiak**, M. A. Jacobs, M. W. Chan, M. L. Comer, and E. J. Delp, "Predicting Hypersonic Glide Vehicle Behavior with Stochastic Grammars", in *IEEE Transactions on Aerospace and Electronics*, 2023, *Submitted for Review.*

Book Chapters Resulting from this Dissertation:

2. H. Hao*, **E. R. Bartusiak***, D. Güera, D. Mas, S. Baireddy, Z. Xiang, S. K. Yarlagadda, R. Shao, J. Horváth, J. Yang, F. Zhu, and E. J. Delp, "Deepfake Detection Using Multiple Data Modalities", in *Handbook of Digital Face Manipulation and Detection - From DeepFakes to Morphing Attacks, Series on Advances in Computer Vision and Pattern Recognition*, Springer, Cham, January 2022, pp. 235-254. DOI: 10.1007/978-3-030-87664-7_11.

Conference Publications Resulting from this Dissertation:

3. **E. R. Bartusiak**, K. Bhagtani, A. K. S. Yadav, and E. J. Delp, "Transformer Ensemble for Synthesized Speech Detection", 2023, *Submitted for Review.*

4. **E. R. Bartusiak**, M. A. Jacobs, C. F. Spells, M. W. Chan, M. L. Comer, and E. J. Delp, "Transfer Learning for Hypersonic Vehicle Trajectory Prediction", *Proceedings of the IEEE Aerospace Conference*, pp. 1-8, March 2023, Big Sky, MT, USA.

5. **E. R. Bartusiak** and E. J. Delp, "Transformer-Based Speech Synthesizer Attribution in an Open Set Scenario", *Proceedings of the IEEE International Conference on Machine Learning and Applications*, pp. 1-8, December 2022, Nassau, The Bahamas. DOI: 10.48550/arXiv.2210.07546.

*denotes equal authorship.

6. **E. R. Bartusiak**, H. Hao, M. A. Jacobs, N. X. Nguyen, M. W. Chan, M. L. Comer, and E. J. Delp, "A Stochastic Grammar Approach to Predict Flight Phases of a Hypersonic Glide Vehicle", *Proceedings of the IEEE Aerospace Conference*, pp. 1-15, March 2022, Big Sky, MT, USA. DOI: 10.1109/AERO53065.2022.9843362.

7. **E. R. Bartusiak** and E. J. Delp, "Synthesized Speech Detection Using Convolutional Transformer-Based Spectrogram Analysis", *Proceedings of the IEEE Asilomar Conference on Signals, Systems, and Computers*, pp. 1426-1430, October 2021, Asilomar, CA, USA (Virtual). DOI: 10.1109/IEEECONF53345.2021.9723142.

8. **E. R. Bartusiak**, N. X. Nguyen, M. W. Chan, M. L. Comer, and E. J. Delp, "A Machine Learning Approach to Classify Hypersonic Vehicle Trajectories", *Proceedings of the IEEE Aerospace Conference*, pp. 1-14, March 2021, Big Sky, MT, USA (Virtual). DOI: 10.1109/AERO50100.2021.9438274.

9. **E. R. Bartusiak** and E. J. Delp, "Frequency Domain-Based Detection of Generated Audio", *Proceedings of the Media Watermarking, Security, and Forensics Conference, IS&T Electronic Imaging Symposium*, pp. 273(1)-273(7), January 2021, Burlingame, CA, USA (Virtual). DOI: 10.48550/arXiv.2205.01806.

Other Publications:

10. K. Bhagtani, **E. R. Bartusiak**, A. K. S. Yadav, P. Bestagini, and E. J. Delp, "Synthesized Speech Attribution Using the Patchout Spectrogram Attribtion Transformer", 2023, *Submitted for Review.*

11. A. K. S. Yadav, Z. Xiang, **E. R. Bartusiak**, P. Bestagini, S. Tubaro, and E. J. Delp, "ASSD: Synthetic Speech Detection in the AAC Compressed Domain", 2023, *Submitted for Review.*

12. A. K. S. Yadav, **E. R. Bartusiak**, K. Bhagtani, and E. J. Delp, "Synthetic Speech Attribution Using Self Supervised Audio Spectrogram Transformer", *Proceedings of the*

Media Watermarking, Security, and Forensics Conference, IS&T Electronic Imaging Symposium, January 2023, San Francisco, CA, USA.

13. **E. R. Bartusiak**, M. Barrabés, A. Rymbekova, J. Gimbernat-Mayol, C. Lopéz, L. Barberis, D. Mas, X. Giró-i-Nieto, and A. G. Ioannidis, "Predicting Dog Phenotypes from Genotypes", *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 3558-3562, July 2022, Glasgow, Scotland, UK. DOI: 10.1109/EMBC48229.2022.9870905.

14. K. Bhagtani, A. K. S. Yadav, **E. R. Bartusiak**, Z. Xiang, R. Shao, S. Baireddy, and E. J. Delp, "An Overview of Recent Work in Multimedia Forensics: Methods and Threats", pp. 1-17, November 2022, *arXiv:2204.12067*. DOI: 10.48550/arXiv.2204.12067.

15. K. Bhagtani, A. K. S. Yadav, **E. R. Bartusiak**, Z. Xiang, R. Shao, S. Baireddy, and E. J. Delp, "An Overview of Recent Work in Multimedia Forensics", *Proceedings of the IEEE International Conference on Multimedia Information Processing and Retrieval*, pp. 324-329, August 2022, San Jose, CA, USA (Virtual). DOI: 10.1109/MIPR54900.2022.00064.

16. E. D. Cannas, S. Baireddy, **E. R. Bartusiak**, S. K. Yarlagadda, D. Mas, P. Bestagini, S. Tubaro, and E. J. Delp, "Open-Set Source Attribution for Panchromatic Satellite Imagery", *Proceedings of the IEEE International Conference on Image Processing*, pp. 3038-3042, September 2021, Anchorage, AK, USA (Virtual). DOI: 10.1109/ICIP42928.2021.9506600.

17. H. Hao, S. Baireddy, **E. R. Bartusiak**, L. Konz, K. LaTourette, M. Gribbons, M. Chan, M. Comer, and E. J. Delp, "An Attention-Based System for Damage Assessment Using Satellite Imagery", *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, pp. 4396-4399, July 2021, Brussels, Belgium (Virtual). DOI: 10.1109/IGARSS47720.2021.9554054.

18. H. Hao, S. Baireddy, **E. R. Bartusiak**, M. Gupta, K. LaTourette, L. Konz, M. Gribbons, M. Chan, M. Comer, and E. J. Delp, "Building Height Estimation via Satellite

Metadata and Shadow Instance Detection", *Proceedings of the SPIE Automatic Target Recognition Conference*, pp. 1-16, April 2021, Orlando, FL, USA (Virtual). DOI: 10.1117/12.2585012.

19. D. Mas, H. Hao, S. K. Yarlagadda, S. Baireddy, R. Shao, J. Horváth, J. Yang, **E. R. Bartusiak**, D. Güera, F. Zhu, and E. J. Delp, "Deepfakes Detection with Automatic Face Weighting", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Workshop on Media Forensics*, pp. 2851-2859, June 2020, Seattle, WA, USA (Virtual). DOI: 10.1109/CVPRW50498.2020.00342.

20. **E. R. Bartusiak**, "An Adversarial Approach to Spliced Forgery Detection and Localization in Satellite Imagery", *Purdue University*, pp. 1-54, May 2019, West Lafayette, IN, USA. DOI: 10.25394/PGS.8035952.v1.

21. **E. R. Bartusiak**, S. K. Yarlagadda, D. Güera, F. Zhu, P. Bestagini, S. Tubaro, and E. J. Delp, "Splicing Detection And Localization In Satellite Imagery Using Conditional GANs", *Proceedings of the IEEE International Conference on Multimedia Information Processing and Retrieval* , pp. 91-96, March 2019, San Jose, CA, USA. DOI: 10.1109/MIPR.2019.00024.