الجمهورية الشعبية الديمقراطية الجزائرية
People's Democratic Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific Research

University of 20 August 1955 – Skikda
جامعة 20 أوت 1955-سكيكدة

THESIS

To obtain the degree of Master's
Field: **Computer Science**
Specialization**: Computer Systems**

THEME

Multimodal Deception Detection Using Machine and Deep Learning:

A Comparative Study on Audio and Visual Cues

Presented by:
CHAGUETMI Ilhem
TADJINE Meram

Submission Date: **June, 2025**
In front of the jury composed of:

Mr. BOUGAMOUZA Fateh	Supervisor
Ms. DAIBOUNE	President
Mrs. ALI GUECHI farida	Examiner

Academic Year: 2024/2025

Acknowledgements

“

We would like to express our sincere gratitude to our supervisor, **Dr. Fateh BOUGAMOUZA**, for his unwavering support, invaluable guidance, and expert knowledge throughout the entire research process. His dedication and mentorship have been instrumental in shaping the direction and quality of this project.

We are also deeply thankful to the faculty members of the Computer Science Department at the University of 20 Aout 1955 – Skikda for their insightful courses, enlightening discussions, and the intellectually stimulating environment they provided. Their expertise and passion for their respective fields have been a constant source of inspiration for us.

”

شكر وتقدير

بسم الله الرحمن الرحيم

قالوا: "العلم يرفع بيوتًا لا عماد لها، والجهل يهدم بيوت العزّ والكرم."

أولاً، شكرٌ لله تعالى على نعمه التي لا تُحصى. لك الحمدُ دائمًا على توفيقك ورحمتك التي منحتنا بها ثمارَ الجدِّ والنجاحِ.

لمشرفي الكريم د. بوقموزا فاتح

شكراً لإرشادك الواضح ونصائحك السديدة التي أنارت دربنا وأوصلتنا إلى برّ الأمان في كل منعطفٍ من هذا البحث.

لزميلتي في البحث إلهام

يا مرآةَ الأفكارِ الصافيةِ التي انعكستْ فيها رؤاي، يا من خضْتِ معي بحورَ البحثِ بجرأةٍ وإصرارٍ دون تردُّدٍ، سطرنا بخطانا المشتركة فصولَ إنجازٍ يعتزُّ به القلبُ، شكراً لكِ على روحِكِ الدّافئةِ وتفانيكِ الذي جعل الرحلةَ أجملَ وأعمقَ.

لوالديّ الغاليين

يا جسرَ الرؤى التي صعدتُ بها إلى أسمى مرام، يا مَن خففتم ثِقَلَ الهمومِ بعطائِكم اللامحدودِ، وبدعائكم الدّائمِ في ليالي السُّهادِ، كنتما السندَ والملاذَ حين تعثّرت خطاي، ولكما معاً في القلبِ منزلةٌ لا تُعاندها كلماتُ الزمانِ.

لسالم

يا من وثق بي حين شكَّ بي الشكُّ، وسندتَني في أحلكِ اللحظات، وأحييتَ في قلبي التفاؤلَ والإيمانَ، واحتفيتَ بأصغرِ إنجازاتي كأنها أعظمُ الانتصارات

شكراً لوقوفكَ الدائمِ بجانبي، ولقلبكَ الذي لا ينطفئُ من الإيمانِ والحبِّ.

لإخوتي إسحاق وتقوى ولأصدقائي وعائلتي

شكراً لدعمكم المستمر وثقتكم بي في كل مرحلة، وشكراً لوجودكم الذي أضاء دروبي وأزال عن قلبي ثِقلَ التحديات، وشكراً لابتساماتكم التي تمنحني القوة وتزرع في روحي بذورَ الأمل.

بحضوركم ازدانت رحلتي وتجلّى في مسعاي سحرُ الوفاء.

And at the end of the day, all I have to say is
ALLAHU AKBAR

Dedication

﴿وَمَا تَوْفِيقِي إِلَّا بِاللَّهِ، عَلَيْهِ تَوَكَّلْتُ وَإِلَيْهِ أُنِيبُ﴾

سورة هود، الآية ٨٨

All praise belongs to Allah, the One who never abandons the hearts that trust in Him. Every step I've taken, every challenge I've overcome, and every word written in this work has only been possible through His grace.

My sincere thanks go to my respected supervisor, Professor Fateh Bougamouza, whose guidance, support, and encouragement have been invaluable. His wisdom and trust were a light along my path.

To my dearest research companion, Meram thank you for walking beside me in this long and winding road. Your strength, kindness, and determination have been an anchor in moments of doubt, and a spark in moments of achievement.

To my beloved father and mother, no words can ever measure what you mean to me. Your endless prayers, sacrifices, and unconditional love are the true foundation of this success. This is as much your victory as it is mine.

To my sweet sister Aya and my supportive brother Yaakoub thank you for your smiles, your words of encouragement, and for always being the warmth I could return to.

And to my dear friends, and everyone who supported me along the way — those who cheered for me, stood by me in silence, or offered their kindness in the smallest ways your presence mattered more than you know.

This work carries a piece of each one of you. Thank you, from the depths of my heart.

Abstract

This thesis explores the use of artificial intelligence for automated deception detection using multimodal data. Traditional lie detection methods—such as polygraph testing and behavioral observation—are limited by subjectivity and poor reliability. In response, this work investigates machine learning and deep learning models applied to audio and visual cues from the Real-Life Trial Deception Dataset (RLDD), which features authentic courtroom testimonies.

The proposed pipeline covers feature extraction, model training, and system deployment. Audio features were extracted using the ComParE_2016 set, while visual features were generated through Vision Transformer (ViT) embeddings. Multiple models, including SVM, XGBoost, Conv1D, BiGRU, and CNN+LSTM, were tested across four configurations: audio-only, visual-only, early fusion, and late fusion.

SVM and Conv1D performed best for audio inputs, while BiGRU and CNN+LSTM achieved the highest accuracy for visual inputs. The final system—LieBusters—uses SVM for audio and BiGRU for visual detection, integrated via decision-level late fusion. It features a real-time interface, recording tools, and LIME-based interpretability for ethical transparency.

This work provides both a practical prototype and a benchmark for future AI-driven behavioral analysis tools.

Keywords: Deception Detection, Machine Learning, Deep Learning, Multimodal Data, Audio-Visual Features, LieBusters, Artificial Intelligence

---

Résumé

Ce mémoire explore l'utilisation de l'intelligence artificielle pour la détection automatisée du mensonge à partir de données multimodales. Les méthodes traditionnelles, comme le polygraphe et l'observation comportementale, présentent des limites en termes de subjectivité et de fiabilité. Ce travail examine les modèles d'apprentissage automatique et profond appliqués à des indices audio et visuels extraits du **Real-Life Trial Deception Dataset (RLDD)**, composé de témoignages authentiques issus de procès réels.

Le pipeline proposé couvre l'extraction de caractéristiques, l'entraînement des modèles et le déploiement du système. Les caractéristiques audio ont été extraites via la configuration **ComParE_2016**, tandis que les caractéristiques visuelles ont été obtenues à l'aide de **Vision Transformer (ViT)**. Plusieurs modèles (SVM, XGBoost, Conv1D, BiGRU, CNN+LSTM) ont été testés sur quatre configurations : audio seul, visuel seul, fusion précoce et fusion tardive.

SVM et Conv1D ont obtenu les meilleures performances sur l'audio, tandis que BiGRU et CNN+LSTM ont atteint les meilleurs résultats pour la vidéo. Le système final, **LieBusters**, utilise SVM pour l'audio et BiGRU pour la vidéo, intégrés via une fusion décisionnelle. Il comprend une interface en temps réel, des outils d'enregistrement et une interprétabilité via LIME pour garantir la transparence.

Mots-clés : Détection de mensonge, Apprentissage automatique, Apprentissage profond, Données multimodales, Audio-visuel, LieBusters, Intelligence artificielle

---

ملخص

يهدف هذا البحث إلى استخدام تقنيات الذكاء الاصطناعي للكشف الآلي عن الكذب باستخدام بيانات متعددة الوسائط. تعاني الطرق التقليدية مثل اختبار جهاز كشف الكذب وملاحظة السلوك من ضعف في الموضوعية والدقة. لذلك، يعتمد هذا العمل على نماذج التعلم الآلي والعميق لتحليل الإشارات الصوتية والبصرية المستخرجة من **قاعدة بيانات محاكمات حقيقية (RLDD)**والتي تحتوي على شهادات واقعية من قاعات المحاكم.

يتضمن النظام المقترح مراحل استخراج الميزات، تدريب النماذج، وتنفيذ النظام. تم استخراج الميزات الصوتية باستخدام مجموعة **ComParE_2016**، بينما تم الحصول على الميزات البصرية باستخدام **Vision Transformer**. تم اختبار عدة نماذج مثلSVM ، XGBoost، Conv1D، BiGRU وCNN+LSTM على أربعة أوضاع: الصوت فقط، الفيديو فقط، الدمج المبكر، والدمج المتأخر.

حقق كل من SVM و Conv1D أفضل أداء في الصوت، بينما تفوق BiGRU و CNN+LSTM في الفيديو. تم اختيار SVM و BiGRU لإنشاء النظام النهائي **LieBusters** باستخدام الدمج المتأخر على مستوى القرار. يتضمن النظام واجهة تفاعلية فورية، وأدوات تسجيل، وآلية تفسير تعتمد على LIME لضمان الشفافية.

الكلمات المفتاحية :كشف الكذب، التعلم الآلي، التعلم العميق، البيانات متعددة الوسائط، الصوت والصورة، LieBusters، الذكاء الاصطناعي

Table of Contents

List of Tables

List of Figures

List of Equations

List of Acronyms

AI – Artificial Intelligence

ML – Machine Learning

DL – Deep Learning

RLDD – Real-Life Trial Deception Dataset

LLD – Low-Level Descriptors

SVM – Support Vector Machine

RF – Random Forest

LR – Logistic Regression

DT – Decision Tree

XGBoost – Extreme Gradient Boosting

MLP – Multilayer Perceptron

CNN – Convolutional Neural Network

RNN – Recurrent Neural Network

LSTM – Long Short-Term Memory

BiLSTM – Bidirectional Long Short-Term Memory

GRU – Gated Recurrent Unit

BiGRU – Bidirectional Gated Recurrent Unit

GCN – Graph Convolutional Network

ViT – Vision Transformer

MFCC – Mel-Frequency Cepstral Coefficients

eGeMAPS – Extended Geneva Minimalistic Acoustic Parameter Set

ComParE_2016 – Computational Paralinguistics Challenge Feature Set 2016

ReLU – Rectified Linear Unit

GELU – Gaussian Error Linear Unit

CV – Computer Vision

NLP – Natural Language Processing

BERT – Bidirectional Encoder Representations from Transformers

SHAP – SHapley Additive exPlanations

LIME – Local Interpretable Model-Agnostic Explanations

FFmpeg – Fast Forward MPEG (Multimedia Framework)

UI – User Interface

API – Application Programming Interface

CSV – Comma-Separated Values

FPS – Frames Per Second

ANN – Artificial Neural Network

VSA – Voice Stress Analysis

FACS – Facial Action Coding System

ICMI – International Conference on Multimodal Interaction

General Introduction

In recent years, the field of deception detection has gained growing interest across academic, industrial, and governmental sectors. Determining whether an individual is being truthful or deceptive is crucial in various domains, including criminal justice, border security, forensic psychology, insurance fraud, and cybersecurity. Traditional methods—such as polygraph examinations, voice stress analysis, and behavioral observation—have been widely used in these contexts. However, they remain controversial due to their dependence on subjective interpretation, vulnerability to countermeasures, and poor generalizability across populations and settings. In an increasingly automated and data-centric world, there is a pressing need for objective, scalable, and reliable alternatives that can manage the complexity of human behavior.

Artificial Intelligence (AI), particularly through machine learning (ML) and deep learning (DL), has emerged as a promising solution in behavioral signal processing. These technologies can automatically extract structured representations from raw multimodal data (e.g., audio, visual), uncover subtle patterns of deception, and provide probabilistic predictions in real-time. By leveraging AI, deception detection systems can become faster, more consistent, and less invasive than traditional approaches.

The central research problem addressed in this thesis is:

“ How can AI-based models be effectively applied to detect deception in real-world conditions using multimodal data? “

Rather than relying on controlled or acted datasets, this work focuses on courtroom testimonies from the Real-Life Trial Deception Dataset (RLDD), which introduces real-world challenges such as data noise, natural expression variability, and small sample size—conditions under which traditional methods often fail.

To address this problem, the thesis sets out the following main objectives:

- To evaluate and compare classical machine learning and modern deep learning models for deception detection;

- To test the discriminative power of audio and visual features both independently and through fusion techniques (early and late fusion);
- To assess the effectiveness of different feature extraction methods, including ComParE_2016, MFCC, Vision Transformer (ViT) embeddings, and BERT-based text embeddings;
- To design and deploy a real-time deception detection system that is technically robust and ethically mindful.

The thesis makes the following key contributions:

1. A comprehensive experimental comparison of ML and DL models (e.g., SVM, XGBoost, Conv1D, BiGRU, CNN+LSTM, ViT, GCN) across single and fused modalities using real-world courtroom data;
2. Demonstration of the superior performance of **late fusion** techniques, where models are trained per modality and combined at the decision level;
3. Evidence supporting the effectiveness of transformer-based architectures (e.g., ViT) and emotionally expressive audio features (ComParE_2016) in deception detection;
4. Implementation of a working prototype system (**LieBusters**) that performs real-time multimodal deception analysis using webcam and microphone input, feature extraction, and feedback display;
5. Integration of **LIME-based interpretability tools** to enhance transparency in model predictions—especially important for ethical use in sensitive applications.

The organization of this thesis is as follows:

- **Chapter I – General Concepts** introduces the foundations of AI, machine learning, and deep learning, and their relevance to deception detection;
- **Chapter II – Lie Detection Methods** presents traditional and modern techniques, available datasets, and a critical review of prior work;
- **Chapter III – Methods and Experiments** outlines the experimental setup, including dataset selection, feature extraction, model training, fusion strategies, and evaluation;
- **Chapter IV – Results and Discussion** analyzes the performance of all tested models, compares features, discusses overfitting and generalization, and synthesizes the key findings;

- **Chapter V – Implementation and Deployment** describes the real-time system architecture, UI interface, feedback mechanisms, and the practical considerations of deploying an AI-driven deception detector.

This thesis seeks to bridge the gap between theoretical model development and practical system deployment. Combining rigorous experimentation with real-time application contributes to the advancement of intelligent behavioral analysis tools and lays a foundation for future research in automated deception detection.

.

Chapter I. State of the Art

I.1. Introduction

Artificial Intelligence (AI) represents one of the most transformative technological advancements of the modern era. Its ability to learn from data, adapt to new information, and automate decision-making processes has redefined the boundaries of what machines can achieve. This chapter presents a structured overview of the fundamental concepts that form the foundation of AI, with a particular emphasis on Machine Learning (ML) and Deep Learning (DL). These two domains have emerged as core drivers of intelligent systems capable of tackling complex problems in diverse domains. Through exploration of learning paradigms, model architecture, training algorithms, and optimization strategies, this chapter establishes the theoretical foundation, necessary for understanding how machines are designed to learn and reason from experience.

I.2. Artificial Intelligence

AI refers to the capability of computational systems to perform tasks typically associated with human intelligence, such as learning, reasoning, problem-solving, perception, and decision-making [1]. This encompasses a range of technologies that enable machines to mimic complex human abilities and interactions [2].

The field of AI was initiated in the 1950s by Alan Turing, who proposed the concept of machine intelligence and the Turing Test [3]. Its formal foundation was established at the 1956 Dartmouth Conference, where AI was recognized as a research discipline [4]. The 1960s–80s were dominated by symbolic AI and expert systems like DENDRAL [5]. In the 1990s, statistical machine learning approaches emerged [6], followed by deep learning breakthroughs in the 2010s. Today, generative models such as ChatGPT exemplify the capabilities of modern AI [7].

I.2.1. Subfields of Artificial Intelligence

Artificial Intelligence (AI) comprises several subfields that collectively aim to replicate various aspects of human intelligence. The most prominent of these are:

- **Machine Learning (ML):** Focuses on designing algorithms that learn patterns from data and make decisions or predictions without being explicitly programmed. ML is

commonly divided into supervised, unsupervised, Semi-Supervised, and reinforcement learning approaches [6].

- **Deep Learning (DL):** A subset of ML that uses multi-layered neural networks to automatically extract complex features from raw data. DL has enabled major advances in image recognition, speech processing, and language understanding [8].
- **Natural Language Processing (NLP):** Enables machines to process and generate human language, supporting tasks such as translation, sentiment analysis, and question answering [9].
- **Computer Vision (CV):** Allows machines to interpret and analyze visual data, such as images and videos. CV is widely used in areas like facial recognition, medical imaging, and autonomous driving [10].

These subfields often operate together in intelligent systems, contributing to applications such as behavior understanding, emotion recognition, and deception detection.

While Computer Vision (CV) is widely recognized as a subfield of Artificial Intelligence, it also incorporates techniques from fields such as signal processing and geometry. The following diagram illustrates the conceptual relationships among major AI subdomains.

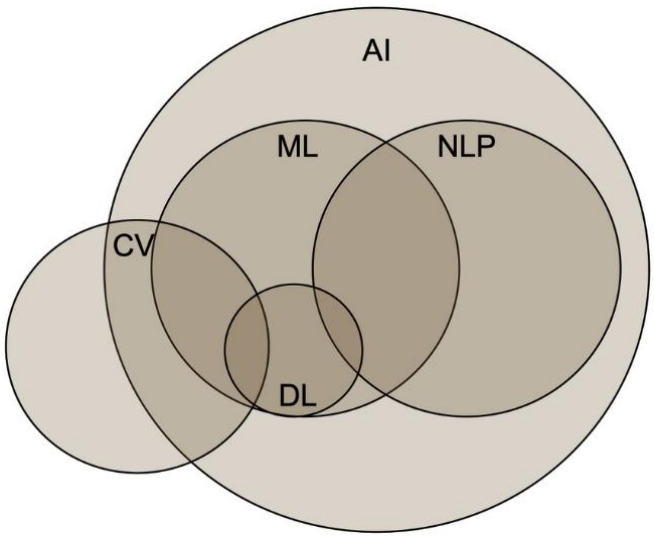


Figure 1 Conceptual map of Artificial Intelligence and its subfields.

I.3. Machine Learning (ML)

ML is a subset of AI that focuses on developing computational models capable of learning from data and improving performance over time without being explicitly

programmed [6]. Instead of following a rigid set of instructions, ML algorithms build mathematical models based on input data, allowing them to make data-driven predictions or decisions in dynamic environments.

ML systems are trained using datasets that contain examples of past experiences. By identifying patterns and statistical regularities within this data, these models can generalize to new, unseen inputs. ML is especially useful when traditional programming is infeasible due to the complexity or variability of the task at hand.

ML is designed to handle a wide variety of data types, including structured (e.g., tabular data), semi-structured (e.g., JSON or XML), and unstructured data (e.g., text, audio, and images). This flexibility makes it highly suitable for real-world applications in healthcare, finance, law enforcement, and social media.

I.3.1. Categories of Machine Learning

ML is commonly divided into four primary categories based on the nature of the data and the level of supervision involved: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning.

Each of these approaches addresses different types of problems and data structures, offering unique strengths and applications. The following subsections provide a detailed overview of each learning category, highlighting their principles, methodologies, and relevance in contemporary ML systems.

I.3.1.1. Supervised Learning

Supervised Learning is a fundamental paradigm within Machine Learning (ML) where models are trained using labeled datasets. Each example in the training data consists of an input vector and an associated output value, often referred to as a ground truth label [6]. The main goal of supervised learning is to approximate a mapping function $f:X \rightarrow Y$ such that the model can accurately predict the output Y for new, unseen inputs X.

Supervised learning problems are generally classified into two major types:

- **Classification**: Tasks where the output variable is categorical. The model aims to assign input data into one of several predefined classes. Examples include image classification, speech recognition, and email spam detection.

- **Regression**: Tasks where the output variable is continuous. Here, the model predicts a real-valued output, such as the prediction of house prices or forecasting stock market trends.

A diverse range of algorithms have been developed for supervised learning, including but not limited to Decision Trees, Support Vector Machines (SVMs), k-Nearest Neighbors (k-NN), Logistic Regression, Random Forests, and Deep Neural Networks [11].

a. **Mechanisms of Learning**: The supervised learning workflow typically involves splitting the dataset into training, validation, and testing sets. During training, the model iteratively adjusts its internal parameters to minimize a loss function, such as cross-entropy loss for classification tasks or mean squared error for regression tasks. Evaluation metrics like accuracy, precision, recall, F1-score, and R-squared (R^2) are employed to assess model performance depending on the nature of the task [12].
b. **Challenges in Supervised Learning**: Despite its popularity, supervised learning comes with inherent challenges:
   - **Label Dependence**: High-quality labeled datasets are essential, and labeling large datasets can be expensive, time-consuming, and prone to human error.
   - **Overfitting**: Models trained on limited or noisy data can memorize the training data rather than generalizing to new data, leading to poor performance on unseen examples.
   - **Bias and Variance Trade-off**: Striking a balance between underfitting (high bias) and overfitting (high variance) is critical for model generalization.
c. **Advances and Trends**: Recent advancements have seen the rise of techniques such as transfer learning, where pre-trained models on large datasets are fine-tuned for specific tasks.

Supervised learning remains one of the most powerful and widely used approaches in machine learning, laying the foundation for numerous modern applications across diverse domains.

The process of supervised learning, including data labeling, model training, and prediction, is illustrated in **Figure 2**.

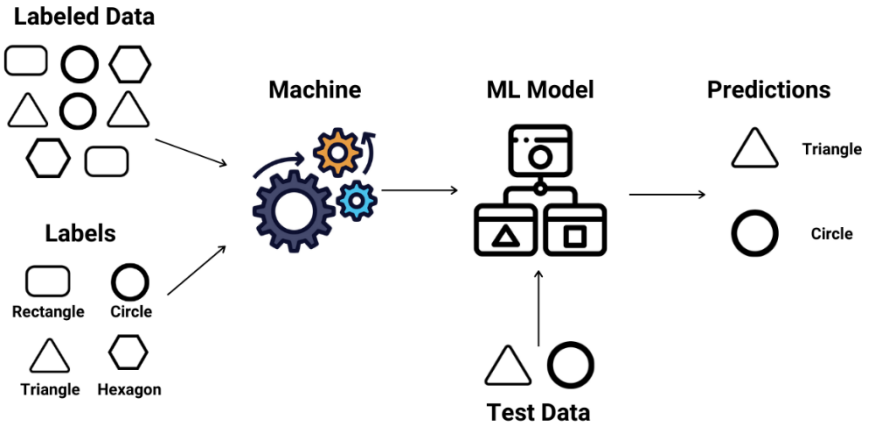


Figure 2 Overview of the Supervised Learning Workflow [13].

I.3.1.2. Unsupervised Learning

Unsupervised Learning is a machine learning approach where models are trained on datasets that do not contain labeled outputs. Instead, the objective is to discover hidden patterns, structures, or relationships within the input data itself [6]. Unlike supervised learning, there is no explicit feedback signal guiding the learning process; the system must infer the underlying organization of the data without predefined categories.

The primary tasks associated with unsupervised learning include:

- **Clustering**: The task of grouping similar data points into clusters based on a defined similarity measure. Popular clustering algorithms include k-Means, DBSCAN (Density-Based Spatial Clustering of Applications with Noise), and Hierarchical Clustering [14].
- **Dimensionality Reduction**: involves minimizing the number of input variables in a dataset while retaining its key structure. Common techniques include **PCA**, **t-SNE**, and **UMAP**, which are used for visualization, noise reduction, and feature extraction [15].
- **Anomaly Detection**: Identifying data points that deviate significantly from the norm, useful in fraud detection, network security, and industrial monitoring.

a. **Mechanisms of Learning:** Unsupervised models often rely on similarity measures such as Euclidean distance, cosine similarity, or probabilistic relationships to identify structure within the data. Clustering algorithms, for instance, partition the dataset into subgroups where intra-group similarity is maximized and inter-group similarity is minimized.
b. **Challenges in Unsupervised Learning**: While unsupervised learning offers powerful tools for data exploration and feature learning, it faces notable challenges:
    - **Evaluation Difficulty**: Without ground truth labels, assessing the quality of the model's output is inherently challenging. Internal evaluation metrics like silhouette score, Davies–Bouldin index, or reconstruction error are often used but may not align perfectly with human intuition.
    - **Model Selection and Interpretability**: Choosing the appropriate number of clusters or the right dimensionality reduction method often requires domain knowledge and iterative experimentation.
    - **Scalability**: Some algorithms struggle with very large datasets, necessitating efficient approximations or distributed learning approaches.
c. **Advances and Trends**: Recent developments in unsupervised learning have been fueled by deep learning methods, leading to models such as autoencoders, variational autoencoders (VAEs), and self-supervised learning frameworks. These techniques enable more effective feature extraction and latent space modeling, often serving as a precursor for supervised tasks [16].

Unsupervised learning remains critical in domains where labeled data is scarce or unavailable, making it an essential component in areas such as bioinformatics, market segmentation, recommender systems, and exploratory data analysis.

As shown in **Figure 3**, unsupervised learning involves processing unlabeled data to uncover hidden patterns or clusters based on similarity.

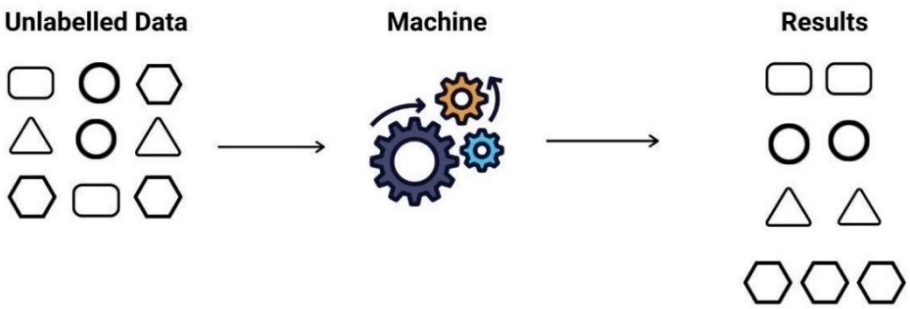


Figure 3 Clustering Process in Unsupervised Learning [13].

I.3.1.3. Semi-Supervised Learning

Semi-supervised learning is a machine learning approach that uses a small amount of labeled data combined with a large quantity of unlabeled data to improve learning accuracy. It bridges the gap between supervised and unsupervised learning, reducing the need for extensive labeled datasets. This method is widely applied in areas where labeling is costly, such as medical diagnosis, natural language processing, and web content classification [17], [18].

As shown in **Figure 4**, semi-supervised learning combines a small set of labeled data with a larger pool of unlabeled data to improve model predictions.

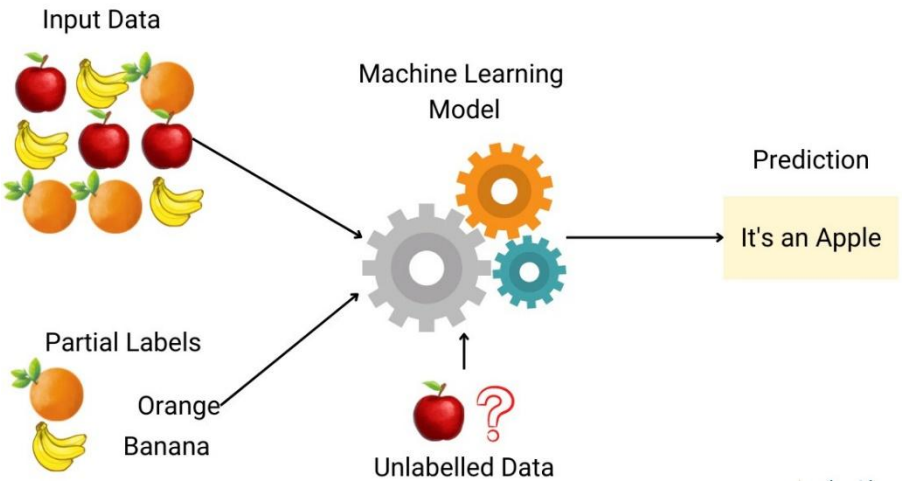


Figure 4 Semi-Supervised Learning with Partially Labeled Data [13].

I.3.1.4. Reinforcement Learning

Reinforcement Learning (RL) is a learning paradigm where an agent learns optimal actions by interacting with an environment to maximize cumulative rewards over time [20]. Based on the framework of Markov Decision Processes (MDPs), the agent observes states, takes actions, and receives rewards to improve its decision-making policy. RL methods are categorized into model-free approaches, which learn directly from experience (e.g., Q-Learning), and model-based approaches, which build a model of the environment. RL has been successfully applied in fields such as robotics, gaming (e.g., AlphaGo), and resource management. [19].

a. Key Components of Reinforcement Learning:

- **Agent**: The learner or decision-maker.
- **Environment**: The external system the agent interacts with.
- **Policy (π\piπ)**: The strategy used by the agent to determine actions.
- **Reward Signal**: The scalar feedback from the environment.
- **Value Function**: Estimates the expected future rewards.
- **Model (optional)**: Predicts the next state and reward.

b. Challenges in Reinforcement Learning:

- **Exploration vs. Exploitation**: Balancing between exploring new actions and exploiting known rewarding actions.
- **Sample Inefficiency**: RL often requires a large number of interactions with the environment.
- **Stability and Convergence**: Learning can be unstable or sensitive to hyperparameters, especially with deep reinforcement learning.

c. Advances and Trends:

Recent advances in RL have been driven by the integration of deep learning, leading to the rise of Deep Reinforcement Learning (DRL). Architectures such as Deep Q-Networks (DQN), Actor-Critic methods, and Proximal Policy Optimization (PPO) have enabled

agents to perform at superhuman levels in complex environments. These developments are expanding RL's application scope into more dynamic, high-dimensional domains [20].

Figure 5 shows how the agent interacts with the environment by taking actions and receiving rewards.

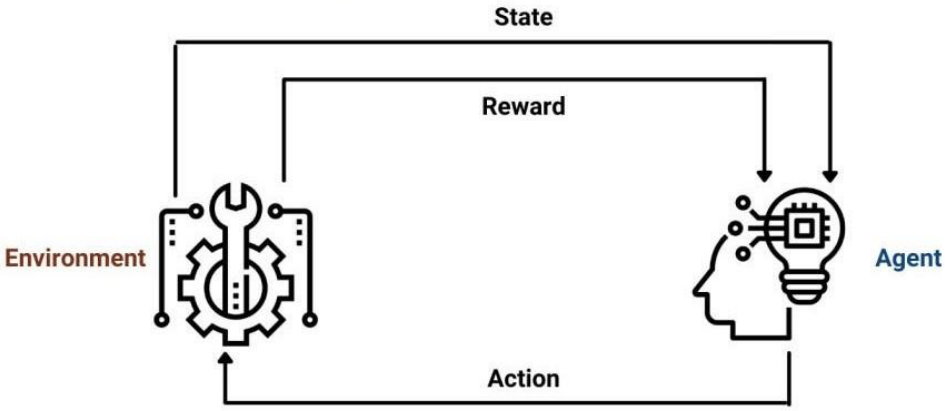


Figure 5 The Agent–Environment Interaction in Reinforcement Learning [13].

I.3.2. Machine Learning Techniques

A variety of machine learning models have been selected in this study for their broad applicability, learning mechanisms, and effectiveness across different data-driven tasks. Each model utilizes a unique strategy to learn from data, supporting diverse approaches to classification, regression, and pattern recognition. The following subsections briefly introduce the models considered.

I.3.2.1. Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised learning algorithm primarily used for binary classification. It works by identifying the optimal hyperplane that maximizes the margin between two classes, where the nearest data points—called support vectors—define the margin [11]. For linearly non-separable data, SVM employs the kernel trick to project inputs into higher-dimensional space using kernel functions such as polynomial, RBF, or sigmoid [21]. To handle noisy or overlapping data, the soft margin approach introduces a regularization parameter C, allowing some misclassifications while

balancing margin size and error minimization [22]. SVMs use hinge loss and solve a convex optimization problem, ensuring a global minimum. They are effective in high-dimensional spaces and require only support vectors for inference, though performance is sensitive to kernel and parameter choices, and training can be resource-intensive on large datasets. As shown in **Figure 6**, SVM constructs an optimal hyperplane that maximizes the margin between classes, with support vectors lying closest to the boundary.

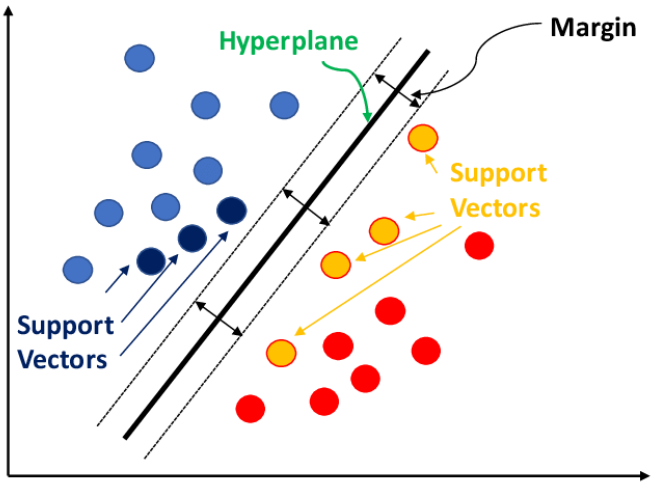


Figure 6 Illustration of SVM Hyperplane, Margin, and Support Vectors [23].

I.3.2.2. Random Forest

Random Forest is an ensemble learning algorithm used for classification and regression tasks. It builds multiple decision trees using bootstrap samples and aggregates their outputs—via majority vote for classification or averaging for regression—to improve prediction accuracy [24]. By randomly selecting subsets of features at each split, it introduces diversity among trees and reduces overfitting. Random Forest is robust, handles high-dimensional and missing data well, and provides insights into feature importance**.** However, it can be computationally intensive and less interpretable than single decision trees. As shown in **Figure 7**, the Random Forest algorithm constructs multiple decision trees from bootstrap samples and combines their outputs through majority voting to produce the final prediction.

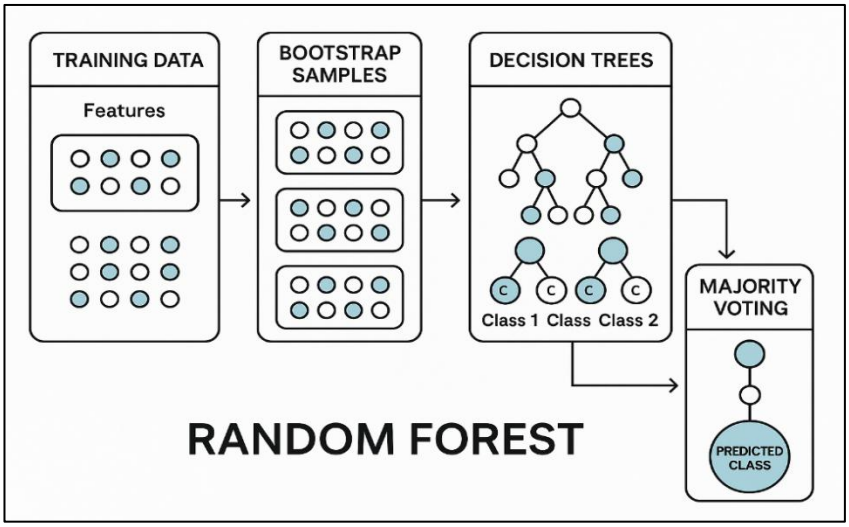


Figure 7 Workflow of the Random Forest Algorithm.

I.3.2.3. Logistic Regression

Logistic Regression is a supervised machine learning algorithm primarily used for binary classification tasks. Unlike linear regression, which predicts continuous outputs, logistic regression models the probability that a given input belongs to a specific class by using the logistic (sigmoid) function [25]. This function maps predictions to a range between 0 and 1, making it ideal for probability-based decision-making.

a. Mathematical Formulation

Logistic Regression models the probability of the positive class as:

$$\hat{y} = \frac{1}{1+e^{-(w^T x+b)}} \quad \ldots\ldots \text{EQ 1}$$

where *w* represents the weight vector, *b* is the bias term, and *x* is the input feature vector. The output $\hat{y}$ represents the predicted probability that the instance belongs to the positive class [11].

The Sigmoid function is used in Logistic Regression to map real-valued inputs to probabilities between 0 and 1. A detailed view of the Sigmoid function and other key activation functions can be found in **Table 1**, which summarizes their formulas and plots.

The model is trained by minimizing **binary cross-entropy loss**:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\left[y^{(i)}\log\left(\hat{y}^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-\hat{y}(i)\right)\right] \quad \ldots\ldots \text{EQ 2}$$

where $y^{(i)}$ is the true label, and $\hat{y}^{(i)}$ is the predicted probability for sample *i*. This loss function encourages the model to produce probabilities that closely match the true labels [11].

I.3.2.4. XGBoost

Gradient Boosting is an ensemble technique where models are trained sequentially, and each new model attempts to correct the errors made by the previous models [26].

The Figure below illustrates the sequential error-correcting process used in Gradient Boosting, where each tree is trained to minimize the residuals of its predecessor.

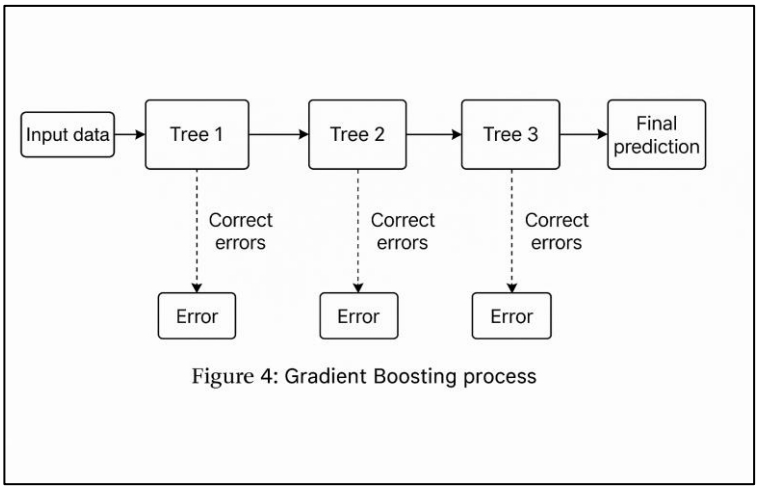


Figure 8 Gradient Boosting Process with Sequential Error Correction.

XGBoost, short for Extreme Gradient Boosting, improves traditional gradient boosting through enhancements in speed, scalability, and accuracy [27]. It introduces key features such as L1/L2 regularization**,** parallelized tree construction**,** sparsity-aware handling**,** and weighted quantile sketch for efficient split finding.

At each iteration, XGBoost minimizes a **regularized objective function**:

$$L = \sum_{i=1}^{n} l(y_i, \hat{y}i) + \sum_{k=1}^{k} \Omega(f_k) \quad \ldots\ldots \text{EQ 3}$$

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \quad \ldots\ldots \text{EQ 4}$$

where:

- $l(y_i, \hat{y}i)$ is the loss function (e.g., log loss).
- T is the number of leaves.
- w_j are the leaf weights.
- γ and λ are regularization hyperparameters.

XGBoost uses a greedy algorithm and a second-order Taylor approximation (gradient and Hessian) to make accurate split decisions, while also handling missing values by learning optimal splitting directions, making it highly robust for sparse data. It is widely recognized for its high predictive accuracy, resilience to overfitting, and efficiency with structured data, though it may require significant computational resources and careful hyperparameter tuning for optimal performance [27].

I.3.3. Advantages and Limitations of Machine Learning

a. Advantages of Machine Learning

- **Works Well with Structured Data:** Machine learning models (like SVM, Random Forest) perform effectively on tabular, numerical, and well-labeled datasets [28].
- **Interpretable Models:** Many ML algorithms, such as Decision Trees and Logistic Regression, provide transparent decision-making processes [29].
- **Requires Less Data than Deep Learning:** Traditional ML models can perform well even on small or medium-sized datasets [30].
- **Faster Training Time:** ML models typically train faster than deep learning models, especially on low-dimensional data.
- **Easier to Implement and Tune:** Simpler architectures and fewer parameters make ML easier to configure and deploy [31].
- **Lower Hardware Requirements:** ML algorithms can run efficiently on standard CPUs without the need for GPUs or large memory.

b. Limitations of Machine Learning

- **Manual Feature Engineering:** ML relies on domain experts to extract relevant features, which can be time-consuming and suboptimal.
- **Limited Performance on Unstructured Data:** ML models struggle with raw image, text, and audio data without extensive preprocessing [12].
- **Scalability Issues:** Some models (e.g., SVMs) become inefficient on very large datasets.
- **Overfitting on Small or Noisy Data:** Without proper regularization or tuning, ML models can overfit to noise.
- **Lower Performance Compared to Deep Learning:** In complex tasks like image recognition or speech processing, ML usually underperforms deep learning [8].

I.4. Deep Learning

DL is a subfield of machine learning that focuses on learning data representations through hierarchical layers of artificial neural networks. These networks are designed to model complex, non-linear patterns in raw data such as images, speech, and text, enabling end-to-end learning without manual feature engineering [8].

While early neural networks date back to the 1940s with the McCulloch-Pitts neuron, practical deep learning became viable in the 2000s with the confluence of three major factors: the availability of large-scale labeled datasets, increases in computing power (especially GPUs), and the development of robust training techniques like backpropagation and gradient descent [32].

Deep learning has led to breakthroughs in domains ranging from image recognition and autonomous vehicles to language modeling. Architectures such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Transformers have become central to these advances. For instance, deep CNNs have achieved human-level accuracy in image classification tasks [33], while Transformer-based models have redefined the state of the art in natural language understanding [34].

Despite these successes, deep learning remains computationally intensive and often requires vast amounts of data, making it less suitable for scenarios with limited resources or interpretability requirements [32].

I.4.1. Fundamentals of Neural Computation

Artificial neural networks (ANNs) are computational models inspired by the structure of biological neurons. They trace their origins to the perceptron, introduced by Frank Rosenblatt in 1958 [41]. While the perceptron could model simple patterns, its inability to solve non-linearly separable problems like XOR led to the development of **multi-layered architectures** with **non-linear activation functions**, forming the foundation of modern deep learning [35].

ANNs mimic biological neurons by processing input signals through interconnected units called **artificial neurons**. Each neuron computes a weighted sum of its inputs, adds a bias, and applies an activation function:

$$y = f(\sum_{i=1}^{n} w_i \, x_i + b) \ldots\ldots \text{EQ 5}$$

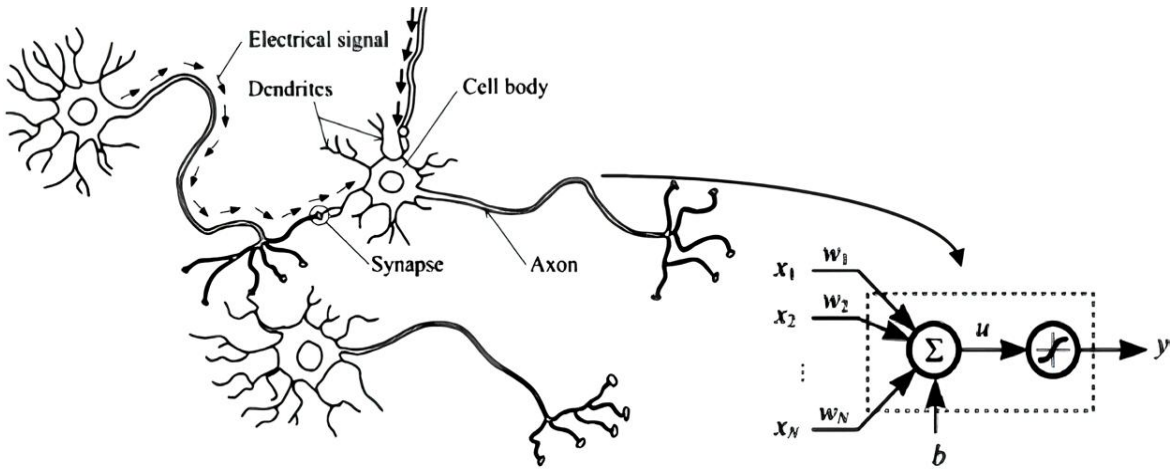


Figure 9 Biological Neuron and Its Artificial Abstraction

Where x_i are the input features, w_i are the corresponding weights, b is the bias, and $f(.)$ is the activation function. This formula forms the building block of all neural network architectures [32].

As illustrated in **Figure 9**, the artificial neuron abstracts the structure of the biological neuron.

a. Activation and Loss Functions in Neural Networks

Activation functions enable neural networks to model **non-linear relationships**. Without them, no matter how many layers are added, the network behaves like a linear model. Below is a table summarizing key activation and loss functions used in deep learning [32].

Table 1 key activation and loss functions.

Sigmoid	**Tanh**	**ReLU**

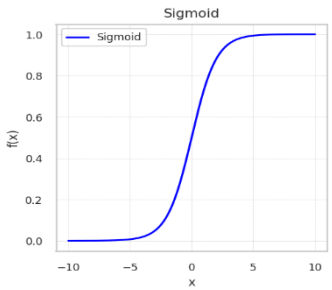
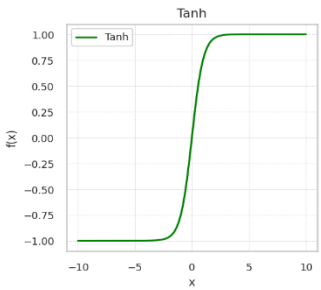
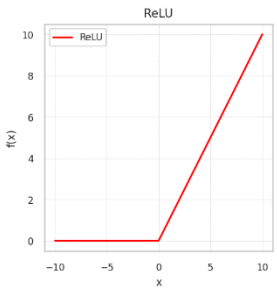

		

Figure 10 Sigmoid Activation Function. $\boldsymbol{\sigma(x) = \frac{1}{1+e^{-x}}}$…… EQ 6	Figure 11 Tanh Activation Function $\boldsymbol{tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}}$…… EQ 7	Figure 12 ReLU Activation Function. $\boldsymbol{ReLU(x) = \max(0, x)}$…..EQ 8
GELU	**Mean Squared Error**	**Cross-Entropy Loss**

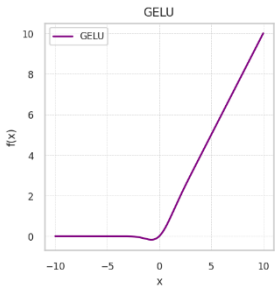
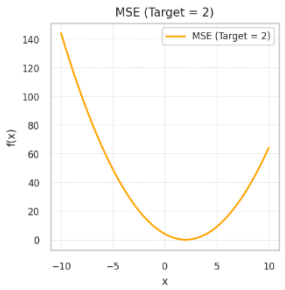
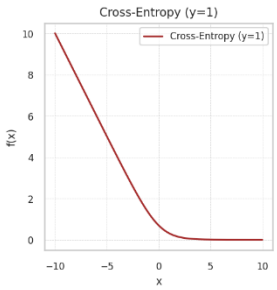

		

Figure 13 GELU Activation Function. $\boldsymbol{GELU(x) = x\,.\,\Phi(x)}$…… EQ 9	Figure 14 Mean Squared Error Loss. $\boldsymbol{MSE = \frac{1}{n}(y_i - \hat{y}i)^2}$……EQ 10	Figure 15 Cross-Entropy Loss. $\boldsymbol{\mathcal{L} = -\sum_i y_i\, log\,(\hat{y}i)}$…… EQ 11

b. Network Architecture

A standard feedforward neural network consists of:

- An input layer that receives raw data,
- One or more hidden layers that transform the data,
- An output layer that produces predictions.

Each layer contains multiple neurons, and the output of each neuron is passed to the next layer, forming a computational graph. These networks are trained to approximate functions and identify patterns in data [32].

c. Training Neural Networks: Backpropagation and Optimization

Neural networks are trained by adjusting their weights to minimize the loss using gradient descent. The gradients are computed using backpropagation, which propagates the error backward through the network:

$$w_i \leftarrow w_i - \eta \cdot \frac{\sigma \mathcal{L}}{\sigma w_i} \quad \ldots\ldots \text{EQ 12}$$

Where η is the learning rate, a hyperparameter that controls the size of updates. More advanced optimizers like Adam, SGD with momentum, and RMSProp improve convergence by adapting learning dynamics during training [32].

I.4.2. Multilayer Perceptron (MLP)

The Multilayer Perceptron (MLP) extends the perceptron by introducing one or more hidden layers between the input and output. These intermediate layers, combined with non-linear activation functions, allow the model to learn complex patterns that single-layer networks cannot capture. Each neuron in an MLP performs a weighted sum of its inputs, adds a bias, and applies an activation function such as ReLU or Tanh.

MLPs are trained using the backpropagation algorithm, which computes the gradient of the loss function with respect to each weight and updates them iteratively using gradient descent [36]. The **Universal Approximation Theorem** demonstrates that a feedforward network with just one hidden layer can approximate any continuous function under mild assumptions, making MLPs theoretically very powerful [37].

While MLPs are suitable for tasks involving structured data, they are less effective for spatial or sequential inputs like images or text, where more specialized architectures such as CNNs or RNNs are preferred [32]. Nevertheless, they remain widely used in classification, regression, and as fully connected decision layers in deeper networks.

I.4.3. Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a class of deep learning models specifically designed to process grid-like data, such as images. Unlike MLPs, which treat all input features equally and lose spatial relationships, CNNs preserve the local structure of data through convolutional layers that apply learnable filters across the input space. These filters slide across the input matrix to detect local patterns such as edges, textures, or shapes [38].

The core components of a CNN include convolutional layers, activation functions (typically ReLU), pooling layers (such as max pooling), and fully connected layers at the end. Convolution layers capture hierarchical features, while pooling layers reduce spatial dimensions, making the model more efficient and less prone to overfitting.

CNNs gained prominence with the success of models like LeNet-5 and later AlexNet, which demonstrated superior performance on large-scale image classification benchmarks like ImageNet [33]. Their applications have since expanded to include medical imaging, object detection, facial recognition, and even audio and time-series classification when adapted to 1D or 3D data.

The core convolution operation of a CNN can be mathematically defined as:

$$h_{i,j}^{(k)} = f\left(\sum_{m=1}^{M}\sum_{p=1}^{P}\sum_{q=1}^{Q} w_{p,q}^{(k,m)} \cdot x_{i+p,j+q}^{(m)} + b^{(k)}\right) \ldots\ldots \text{EQ 13}$$

This formulation describes how input features are combined with learned filter weights to produce each output activation. Here, $f(\cdot)$ represents the activation function (e.g., ReLU), and the sums iterate over input channels and filter dimensions.

Due to their parameter-sharing and local connectivity properties, CNNs are far more efficient than MLPs for processing high-dimensional inputs like images. They have become the standard architecture for most computer vision tasks and are often used as

feature extractors in multimodal or hybrid deep learning systems [32]. The general structure of a CNN is shown in **Figure 16**:

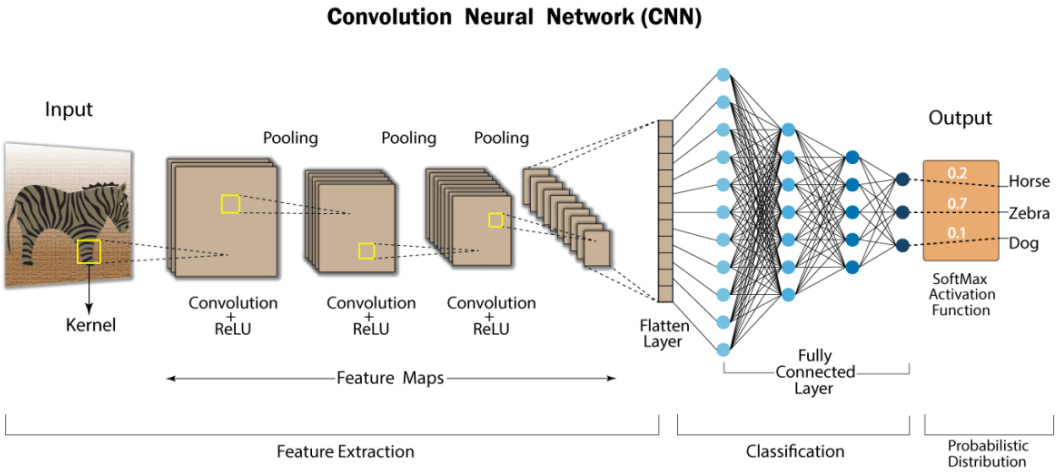


Figure 16 A schematic overview of a Convolutional neural Network (CNN) [39].

I.4.4. Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are designed for modeling sequential or temporal data, making them ideal for applications such as speech recognition, time-series forecasting, and natural language processing. Unlike feedforward networks like MLPs and CNNs, RNNs incorporate recurrence — meaning the output from previous time steps is fed back into the network as input for the current step. This gives the model a form of memory, enabling it to learn dependencies over sequences [40].

The core mechanism of an RNN involves maintaining a hidden state, which is updated at each time step based on the current input and the previous state. This is mathematically defined as:

$$h_t = f(w_{xh}x_t + w_{hh}h_{t-1} + b_h) \ldots\ldots \text{EQ 14}$$

Where h_t is the current hidden state, x_t is the current input, w_{xh} and w_{hh} are weight matrices, and f is a non-linear activation function (commonly tanh or ReLU).

However, traditional RNNs suffer from the vanishing gradient problem, which limits their ability to learn long-term dependencies. To overcome this, advanced variants such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) were

developed. These architectures use gating mechanisms to better control information flow and preserve context across longer sequences [41].

Despite being gradually replaced in some domains by attention-based models like Transformers, RNNs remain valuable for tasks where data is inherently sequential and when computational efficiency is a priority [32].

Figure 17 illustrates the unrolled architecture of a Recurrent Neural Network (RNN), showing the flow of data through input, hidden, and output layers across time steps.

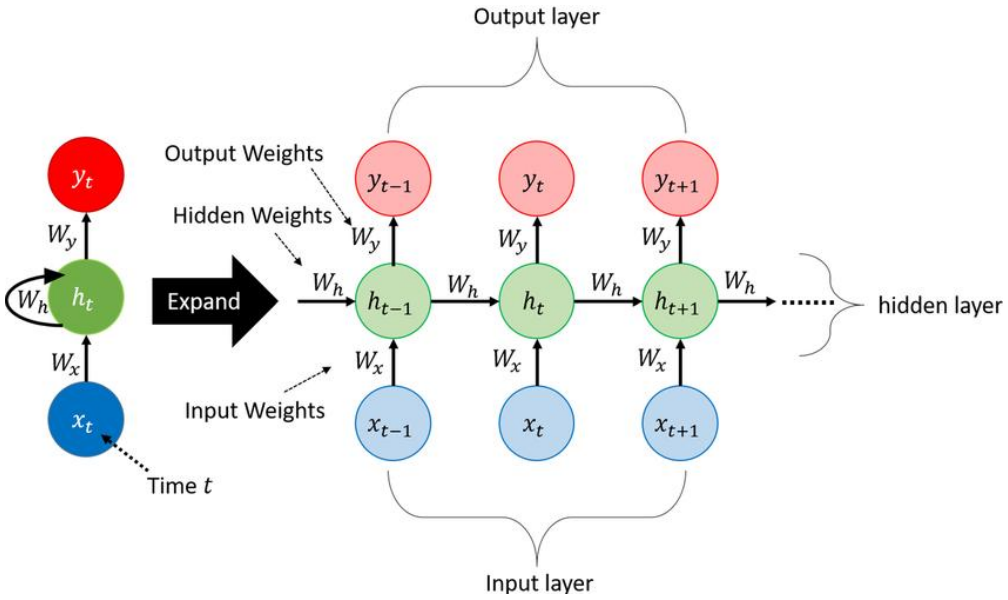


Figure 17 Unfolded architecture of a Recurrent Neural Network (RNN) [42].

I.4.5. Transformer Networks

Transformer networks represent a major breakthrough in deep learning, especially in the field of natural language processing (NLP). Unlike recurrent architectures that process sequences step-by-step, Transformers leverage self-attention mechanisms to model dependencies between all input positions simultaneously, enabling significantly better parallelization and the ability to capture long-range relationships [43].

Introduced by Vaswani et al. in 2017, the original Transformer architecture consists of an encoder–decoder structure. The encoder processes the input sequence into a set of continuous representations, while the decoder generates the output sequence of one element at a time using both the encoded input and previous decoder outputs. The core of

this design is the multi-head self-attention mechanism, which allows the model to weigh the importance of different words or tokens in a sequence, regardless of their distance [43].

The attention mechanism is mathematically defined as:

$$\text{Attention}(\text{Q}, \text{K}, \text{V}) = \text{softmax}\left(\frac{\text{QK}^{\text{T}}}{\sqrt{\text{d}_\text{k}}}\right)\text{V} \quad \ldots\ldots \text{EQ 15}$$

Where Q, K, and V are the query, key, and value matrices, and d_k is the dimensionality of the keys.

Transformers have been the foundation for many modern large-scale models, such as BERT, GPT, and T5, which achieve state-of-the-art performance in tasks like translation, summarization, question answering, and text generation. The Transformer architecture has also been successfully adapted to vision (ViT), audio, and multimodal tasks, reinforcing its versatility [44].

Figure 18 illustrates the Transformer architecture, highlighting its encoder-decoder structure with multi-head attention and feedforward layers.

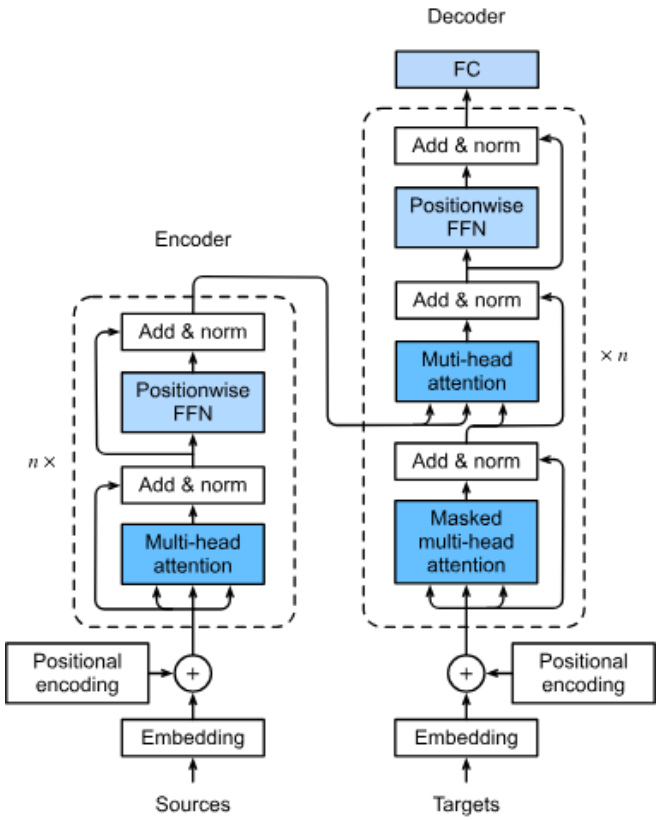


Figure 18 Standard encoder-decoder architecture of a Transformer network [43].

I.4.6. Advantages and Limitations of Deep Learning

a. Advantages of Deep Learning

- **Automatic Feature Extraction**: Learns hierarchical representations from raw data without the need for manual feature engineering [32].
- **End-to-End Learning**: Integrates feature extraction and prediction in a single pipeline, reducing dependency on domain expertise.
- **High Performance on Unstructured Data**: Excels at tasks involving images, text, audio, and video.
- **Scalability with Data**: Performance often improves as more training data becomes available.
- **Transfer Learning**: Pre-trained models can be adapted to new tasks with limited data.
- **State-of-the-Art Results**: Consistently outperforms traditional ML in domains like computer vision, speech recognition, and NLP [32].

b. Limitations of Deep Learning

- **Data Hunger**: Requires large, labeled datasets to achieve high accuracy.
- **High Computational Cost**: Demands powerful hardware (e.g., GPUs, TPUs) and long training times.
- **Black Box Nature**: Lacks interpretability, making it difficult to understand decision processes [45].
- **Overfitting Risk**: Susceptible to memorizing training data, especially on small datasets.
- **Hyperparameter Sensitivity**: Performance depends heavily on proper tuning of architecture and training parameters.
- **Vulnerability to Adversarial Attacks**: Small input changes can cause incorrect predictions in many deep models.

I.5. Comparison Between Machine Learning and Deep Learning

Machine learning and deep learning are closely related fields, with the latter emerging as a powerful extension of the former. While both aim to create models that learn from data, their approaches, capabilities, and requirements differ significantly. The table below highlights the key distinctions between traditional machine learning and deep learning [46].

Table 2 Deep Learning vs. Machine Learning

Aspect	Machine Learning (ML)	Deep Learning (DL)
Definition	Algorithms that learn from data using hand-crafted features	Subset of ML that uses neural networks to learn representations automatically
Data Requirement	Works well on small to medium datasets	Requires large amounts of labeled data
Feature Engineering	Manual feature extraction is often necessary	Learns features automatically from raw data
Performance	Good for structured/tabular data	Superior for high dimensional / unstructured data (e.g. images, text, audio)
Interpretability	High (models like decision trees are easy to interpret)	Low (often seen as "black box" models)
Training Time	Fast and efficient	Slower, more computationally intensive
Hardware Needs	Can run on CPU	Typically requires GPU or TPUs

Deep learning has outperformed traditional machine learning in many fields, especially those involving perceptual and sequential data. However, its limitations—such as the need for large datasets, high computational costs, and low interpretability—still

make classical ML models more suitable in scenarios where data is limited, transparency is important, or real-time computation is critical [45].

I.6. Conclusion

In this chapter, we have surveyed the foundational pillars of Artificial Intelligence, including its most prominent subfields—Machine Learning and Deep Learning. We explored various algorithms, from traditional models like Support Vector Machines and Decision Trees to more expressive architectures such as Multilayer Perceptron's, Convolutional Neural Networks, Recurrent Neural Networks, and Transformers. Special emphasis was placed on the mathematical and functional elements of these models, including activation functions, loss metrics, and training procedures. This theoretical exploration not only clarifies how machines learn from data but also prepares us to understand how such models can be tailored to interpret subtle human behaviors. These insights directly inform the next chapter, where AI techniques are examined in the context of one of the most elusive behavioral challenges—detecting deception.

Chapter II. Lie Detection Methods

II.1. Introduction

Deception detection is a compelling area of research that lies at the intersection of psychology, neuroscience, and artificial intelligence. From a societal standpoint, the ability to distinguish between truthful and deceptive behavior has significant implications for law enforcement, security screening, judicial processes, and even human-computer interaction. This chapter presents a comprehensive overview of lie detection methodologies, beginning with traditional techniques such as the polygraph, voice stress analysis, and body language interpretation. While historically important, these methods often suffer from subjectivity and limited scalability. In response, recent advancements in AI have enabled more objective, data-driven solutions. We explore how machine learning and deep learning models are applied to detect deception through multimodal signals—such as speech and facial expressions—highlighting the field's evolution from expert judgment to algorithmic inference.

II.2. Lie Detection

II.2.1. Definition and Historical Background

Lie detection refers to the process of determining whether an individual is being truthful, typically by analyzing verbal, non-verbal, or physiological signals. Historically, the earliest formal technique was the polygraph, developed in the early 20th century, which aimed to infer deception based on autonomic responses like heart rate and skin conductance [47]. Over time, the field evolved with contributions from psychology and behavioral science, eventually integrating computational approaches. In recent decades, artificial intelligence, especially machine learning and deep learning—has emerged as a transformative force, enabling automatic detection of deception using data from speech, facial expressions, and body language [48].

II.2.2. Importance and Application Areas

The ability to detect deception has critical implications in domains such as criminal justice, national security, forensic psychology, insurance fraud detection, and human-computer interaction. Accurate lie detection tools can assist law enforcement during

interrogations, support border security, or flag suspicious behavior in online communications. AI-based systems offer advantages such as scalability, consistency, and the ability to process multimodal data in real time. However, their deployment also raises concerns regarding ethics, data privacy, and potential biases, making responsible use a vital consideration [49], [50] .

II.3. Traditional Lie Detection Methods

II.3.1. Polygraph Test

The polygraph measures physiological responses such as heart rate, respiration, blood pressure, and skin conductivity to detect deception [47]. It assumes that lying produces stress-related changes distinguishable from truthful behavior. However, emotions like fear or anxiety can mimic these responses, making results difficult to interpret [51]. Moreover, subjects may use physical or mental countermeasures to manipulate the outcome [52]. Due to these limitations, the polygraph's scientific validity is debated, and it is often inadmissible in court. **Figure 19** illustrates a standard polygraph setup used to monitor physiological signals during lie detection assessments.

Figure 19 Polygraph examination setup with physiological signal monitoring.

II.3.2. Facial Expression Analysis

Facial expression analysis detects deception by interpreting involuntary micro-expressions that may reveal hidden emotions like fear or contempt [53]. The Facial Action

Coding System (FACS), developed by Ekman and Friesen, categorizes facial muscle movements into action units for analysis [54]. Trained experts use these cues to infer emotional states during lie detection. However, the method's effectiveness depends on observer skill, cultural variation, and individual facial control. External factors like lighting and bias can also impact accuracy and reliability [55]. Figure 20 presents an overview of facial Action Units as defined by the Facial Action Coding System (FACS), illustrating specific muscle movements used to decode facial expressions during lie detection.

Figure 20 Facial Action Coding System (FACS) [54].

II.3.3. Voice Stress Analysis (VSA)

Voice Stress Analysis (VSA) detects deception by analyzing vocal changes like pitch, tremors, and tone, which are believed to reflect psychological stress [56]. It uses software to examine speech recordings for subtle frequency or amplitude shifts linked to vocal cord tension. Unlike polygraphs, VSA is non-invasive and can be administered remotely [57]. However, its accuracy is challenged by factors such as emotion, fatigue, or noise. Due to its inconsistent reliability and lack of standardization, VSA is typically used as a supplementary tool rather than a standalone method [58]. **Figure 21** displays a spectrographic representation of a speech sample, highlighting frequency and amplitude

changes typically analyzed in Voice Stress Analysis (VSA) to detect vocal indicators of psychological stress.

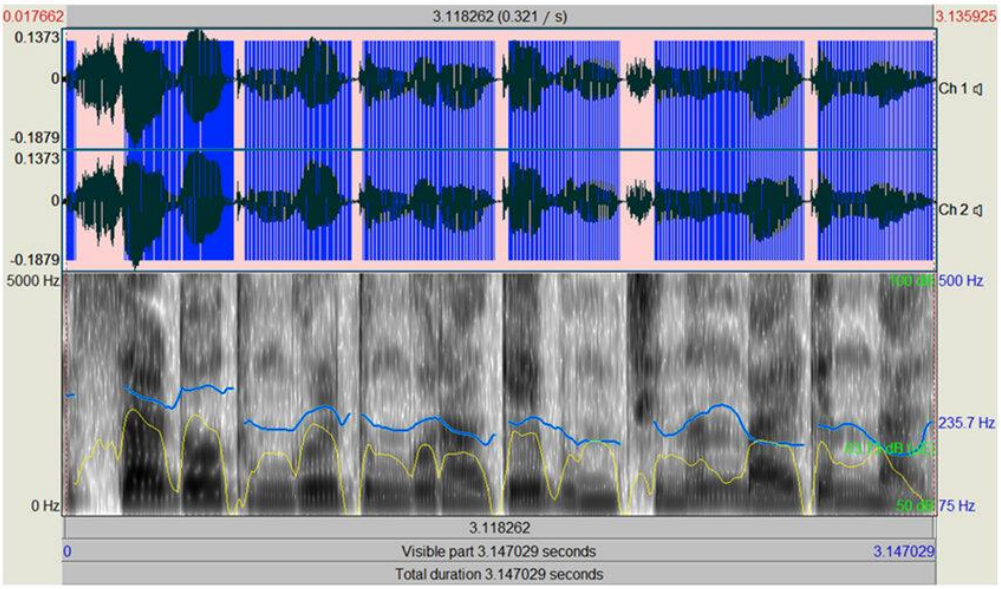


Figure 21 Voice signal waveform used in traditional Voice Stress Analysis (VSA) [59].

II.3.4. Non-verbal Behavior and Body Language

Non-verbal behavior analysis detects deception through involuntary cues like posture shifts, gaze aversion, and gestures linked to psychological discomfort [53]. Based in behavioral psychology, it assumes lying causes stress that manifests in subtle motor expressions [60]. Trained observers assess inconsistencies in movement and timing to spot dishonesty. However, cultural norms and individual traits can affect cue interpretation. Due to these limitations, this method is considered supportive rather than definitive in lie detection [61].

II.3.5. Advantages and Limitations of Traditional Methods

While traditional lie detection methods have been used for decades in forensic, legal, and psychological contexts, their effectiveness and reliability remain debated. It is therefore important to examine both their strengths and shortcomings to understand their current role and the motivation behind transitioning to more modern, AI-based approaches.

a. Advantages

- **Widely used and accessible:** Traditional methods like polygraph and body language analysis are commonly used in law enforcement and interviews [60].
- **Do not require advanced computational systems:** These techniques can often be applied without digital tools or complex models [60].
- **Offer multi-channel input:** Polygraph and facial analysis combine physiological and behavioral data (e.g., heart rate, facial movement) [47].
- **Useful as supportive tools:** Can guide further questioning or raise suspicion during investigations [60].
- **Grounded in psychological theory:** Based on long-standing principles from behavioral science and forensic psychology [61].

b. Limitations

- **Subjective interpretation:** Accuracy often depends on the observer's experience and situational context [60].
- **Susceptible to false positives/negatives:** Anxiety, cultural norms, or physical conditions may mimic signs of deception [61].
- **Scientific reliability is debated:** Studies show inconsistent results, particularly for polygraph and VSA .
- **Limited legal admissibility:** Many courts exclude polygraph results due to lack of standardization .
- **Cannot generalize across individuals:** High variability in how people display stress or deception cues [62].

II.4. AI-based Lie Detection Methods

In recent years, artificial intelligence (AI) has transformed the field of deception detection by introducing automated systems that analyze behavioral, vocal, and physiological cues. These systems use machine learning (ML) and deep learning (DL) algorithms to learn from data and detect complex patterns associated with deceptive behavior. Unlike traditional techniques, AI-based approaches offer increased objectivity, consistency, and potential for real-time processing. This section presents the principles behind ML and DL-based methods and explains how they are applied to the task of lie detection.

II.4.1. Machine Learning Approaches

Machine learning (ML) techniques are widely used in lie detection systems due to their ability to learn from labeled data and identify hidden patterns that may indicate deception. These approaches typically follow a pipeline that includes data preprocessing, feature extraction, model training, and evaluation. ML models can handle various data modalities—such as audio, video, and physiological signals—and are especially effective when domain-relevant features are properly extracted. The following subsections detail the feature extraction process and the most commonly used ML algorithms in deception detection.

II.4.1.1. Feature Extraction in Machine Learning

In machine learning-based lie detection, the first and most crucial step is feature extraction—the process of converting raw behavioral data into measurable variables that a model can analyze. Depending on the modality (audio or visual), different types of features are extracted:

- **Audio features**: These include pitch, jitter, energy, speech rate, and Mel-Frequency Cepstral Coefficients (MFCCs), which capture variations in vocal tone and delivery often associated with stress or deception [56].
- **Visual features**: These involve tracking facial landmarks, head movements, gaze direction, and micro-expressions using facial recognition frameworks or keypoint detection techniques [55].

The goal of this step is to convert multimodal signals into structured vectors or time-series data that can be fed into a machine learning classifier. Proper feature extraction is vital for improving model accuracy and generalization.

II.4.1.2. Common Machine Learning Algorithms

Once features are extracted from audio, visual, or physiological data, various machine learning algorithms can be used to classify behavior as deceptive or truthful. The following are among the most applied models in lie detection research:

a) **Support Vector Machine (SVM)**: Known for its strong performance in high-dimensional spaces and its suitability for small datasets [47].

b) **Random Forest**: A robust ensemble method capable of handling noisy data and providing insights into feature relevance [24].
c) **Logistic Regression**: Serves as a reliable baseline when the relationship between features and labels is relatively linear [25].
d) **Gradient Boosting (e.g., XGBoost)**: Offers high predictive accuracy through sequential model optimization, especially in structured feature spaces [27].

Each model requires appropriate tuning of hyperparameters and validation techniques such as cross-validation to achieve optimal performance in lie detection scenarios.

II.4.2. Deep Learning Approaches

Deep learning (DL) has emerged as a powerful extension of machine learning, capable of automatically learning complex patterns from raw data without manual feature engineering. Unlike traditional ML algorithms, which rely heavily on carefully crafted input features, DL models use multi-layered neural networks to extract hierarchical representations directly from audio, video, or multimodal signals. This makes them particularly effective for processing unstructured data and detecting subtle temporal or spatial cues linked to deception. The following subsections introduce the foundations of deep learning in lie detection and the most used architectures.

II.4.2.1. Fundamentals of Neural Networks for Lie Detection

In lie detection tasks, neural networks are employed to learn subtle and complex patterns embedded in behavioral signals. These models are well-suited for capturing non-linear relationships that span over time (e.g., vocal tension, blinking rate) and space (e.g., facial asymmetry or head pose) [64], [65]. Instead of manually selecting features, neural networks learn useful representations directly from raw or lightly preprocessed inputs such as audio signals or video frames [36].

While the internal structure of neural networks varies depending on the task, most consist of:

- **An input layer** that receives modality-specific data (e.g., audio features, visual frames).
- **One or more hidden layers**, which learn abstract representations from the input.

- **An output layer** that produces a prediction (e.g., truthful or deceptive) based on learned patterns [36].

These networks are trained using labeled data, with parameters updated through backpropagation to minimize prediction error [3]. Once trained, they can generalize to unseen samples and detect deception cues that may not be easily observed by humans, [65].

II.4.2.2. Common Deep Learning Architectures Used in Lie Detection

Deep learning models have become central to deception detection due to their ability to learn complex, non-linear, and multi-modal behavioral patterns. The following architectures are frequently used to analyze signals such as voice, facial movement, body posture, and physiological activity. Each plays a unique role in uncovering hidden indicators of deceit.

a. 1D Convolutional Neural Network (Conv1D)

Conv1D networks apply convolutional filters over one-dimensional sequences, making them ideal for processing audio signals such as raw waveforms or time-series features like MFCCs [66].

- In lie detection, Conv1D captures short-term prosodic and acoustic changes—such as vocal tremors, pitch instability, and silent pauses—that correlate with psychological stress or cognitive load.
- Their low complexity and high speed make them ideal for real-time or resource-limited deception detection setups.
- Conv1D is particularly valuable when working with speech-based datasets, where truthful and deceptive utterances differ in rhythm and tone.

b. Long Short-Term Memory (LSTM)

LSTMs are recurrent neural networks designed to remember long-term dependencies in sequences using specialized memory cells and gates [40].

- In deception contexts, LSTMs can track evolving behavioral cues over an entire utterance or video segment.

- In deception contexts, LSTMs can track evolving behavioral cues over an entire utterance or video segment.
- LSTMs are fundamental when the deception unfolds gradually, especially in long interviews or multi-turn conversations.

c. Bidirectional LSTM (BiLSTM)

BiLSTM process input in both forward and backward directions, allowing access to past and future context simultaneously [67].

- This is particularly useful in lie detection, where the meaning of a behavioral signal often depends on what precedes and follows it (e.g., delayed response after a direct question).
- BiLSTM helps improve classification accuracy by fully modeling the temporal dynamics of speech and gestures.
- They are widely used in detecting speech inconsistencies and context-dependent non-verbal responses.

d. Gated Recurrent Units (GRU and BiGRU)

GRUs are simplified alternatives to LSTMs that achieve similar performance with fewer parameters. BiGRUs combines forward and backward processing [68].

- GRUs are effective in situations with limited data or where fast convergence is needed.
- In lie detection, BiGRUs are especially useful for detecting inconsistencies in vocal tone or gesture flow, offering context-aware sequence modeling.
- Their simplicity and effectiveness make them a popular choice for lightweight deception analysis systems.

e. 2D and 3D Convolutional Neural Networks (CNN, C3D)

2D CNNs extract spatial features from static images or single video frames, while 3D CNNs capture both spatial and temporal information across consecutive frames [69].

- In deception detection, 2D CNNs are often applied to facial imagery to detect emotion leakage or micro-expressions such as brief frowns, smirks, or eyebrow raises.
- 3D CNNs are capable of modeling how these expressions change over time, enabling the system to capture motion-based indicators like body posture shifts or facial stiffness
- Their ability to localize deception-related signals in space and time makes them essential in visual lie detection.

f. Vision Transformers (ViT)

ViT use self-attention mechanisms to process visual data without relying on convolution. They treat image patches as sequences, like words in a sentence [70].

- ViT divides an image into patches and analyzes spatial relationships globally.
- In deception detection, ViT help identify distributed facial indicators across regions that may not be picked up by localized CNN filters.
- Their global attention capability makes them suitable for detecting high-level visual inconsistencies in face or body language [70].

g. Graph Convolutional Networks (GCN, ST-GCN)

GCNs process data represented as graphs, where nodes and edges represent elements like facial landmarks or skeletal joints [71].

- GCNs are ideal for modeling structured facial behavior, such as the coordinated movement of eyebrows, eyes, and mouth during deceptive expressions.
- ST-GCNs (Spatio-Temporal GCNs) extend this idea by modeling how these relationships evolve across time—capturing patterns like unnatural head movement, shoulder tension, or hand gestures.
- Their structured approach provides robustness to noise and a deeper understanding of coordinated deception behaviors.

h. Text-Based Transformers (BERT, GPT)

BERT and GPT are pre-trained transformer models designed for natural language understanding and generation, respectively [72].

- BERT is applied to deception detection by analyzing the **contextual meaning** of words in transcribed speech or written statements, detecting linguistic cues like evasiveness, hedging, or overly formal language.
- GPT models are used to evaluate **semantic coherence**, response fluency, and emotional tone, and may also be fine-tuned to detect manipulative language.
- These models are most suitable when the input modality involves language-based deception (e.g., court transcripts, emails, or interviews).

i. Combined CNN+LSTM Architectures

These models combine the spatial analysis capabilities of CNNs with the temporal modeling power of LSTMs [73].

- CNNs first extract spatial features from visual inputs (e.g., eyes, mouth) or spectrograms, while LSTMs model how these features evolve across time.
- This is crucial in lie detection where both **what** happens (e.g., a micro-smile) and **when** it happens (e.g., during a denial) are equally important.
- The CNN+LSTM architecture is well-suited for multimodal inputs, making it highly effective for analyzing synchronized audio-visual data.

II.5. State of the Art in Lie Detection Research

II.5.1. Recent Research Findings and Models

In recent years, lie detection research has increasingly incorporated advanced artificial intelligence methods, particularly those based on multimodal data and deep learning. A notable trend in the field is the integration of audio, visual, and textual inputs to more accurately detect deceptive behavior.

Pérez-Rosas et al. [74] conducted one of the earliest influential studies using **real-life courtroom trial videos**. Their system extracted features from **speech, facial expressions, and text transcripts**, combining them through traditional machine learning models like **Support Vector Machines (SVM)** and **Random Forests**. Despite the complexity of the real-world data, the model achieved an accuracy of over 75%, demonstrating the promise of multimodal deception detection outside of controlled lab settings.

More recently, King and Neal [49] published a **systematic review in 2024**, analyzing a wide range of AI-enabled lie detection systems using **video, audio, and physiological data**. Their findings emphasized the **superiority of hybrid models** that combine visual and vocal features. They also highlighted the growing use of deep neural networks, particularly **CNNs for facial analysis** and **RNNs (including LSTMs and GRUs) for speech and timing-related patterns**.

Another significant contribution was made by Bahaa et al. [71], who proposed a **multimodal deep learning pipeline** combining **Convolutional Neural Networks (CNN)** with **Long Short-Term Memory (LSTM)** networks. Their model was trained on a combination of audio spectrograms and visual frames, achieving **over 90% classification accuracy**. Their study emphasized the importance of capturing both spatial (facial features, frame-level cues) and temporal (speech rhythm, movement sequences) aspects of behavior.

In terms of architectural advancements, the use of **Vision Transformers (ViT)** and **Graph Convolutional Networks (GCNs)** has gained attention. Dosovitskiy et al. [70] introduced ViTs for image classification and their application in deception detection has been explored due to their ability to analyze **facial micro-expressions** across image patches. Similarly, Yan et al. [71] demonstrated that **Spatial-Temporal Graph Convolutional Networks (ST-GCNs)** are effective in capturing relational dynamics between body joints, particularly useful for modeling full-body movements in video-based deception detection.

These studies reflect a clear shift toward **multimodal, data-driven systems** that go beyond simple physiological measurements. The integration of deep learning and multimodal fusion has not only improved accuracy but also enabled more **scalable, real-time, and non-invasive** solutions.

Beyond conventional audio-visual approaches, recent research has expanded into alternative modalities and advanced models. For example, Large Language Models (LLMs) such as FLAN-T5 have been fine-tuned for verbal lie detection, using datasets containing personal opinions and autobiographical statements to outperform traditional classifiers like logistic regression [75]. Eye-tracking studies have also shown promise, with models like XGBoost achieving up to 74% accuracy in binary deception classification using features such as saccades, fixations, and pupil dilation [76]. These emerging directions reflect the growing interest in leveraging subtle behavioral signals and pre-trained neural architectures to improve deception detection performance in more naturalistic settings.

Table 3 Summary of Recent Research in AI-Based Lie Detection

Study & Year	Modality & Dataset	Models Used	Accuracy	Key Notes
Pérez-Rosas et al. (2015) [21]	Audio, Video, Text – RLDD	SVM, Random Forest	60–75%	Real-life courtroom videos
King & Neal (2024) [67]	Multimodal (Audio, Video, Physio) – Various	CNN, RNN (Review)	Not applicable (N/A)	Hybrid models outperform unimodal
Bahaa et al. (2024) [90]	Audio, Video – Multimodal Dataset	CNN + LSTM	99%	Temporal-spatial fusion
Dosovitskiy et al. (2021) [92]	Visual – ImageNet (Pretrained)	Vision Transformer	88.55%	Sensitive to facial micro-expressions
Yan et al. (2018) [91]	Motion – NTU RGB+D	ST-GCN	81.5–88.3%	Models spatial and temporal joint movement
Chang et al. (2023) [93]	Text – 3 English datasets	FLAN-T5 (LLM)	79.3%	LLM used for verbal deception
Lang & Stricker (2024) [94]	Eye Tracking – Eyelink 1000	XGBoost	74%	Fixations, saccades, pupil features

II.5.2. Key Contributions and Results from Literature

The development of automated lie detection systems has accelerated in recent years, with researchers focusing on how to best utilize machine learning and deep learning

across various modalities. While early work explored simple classification on handcrafted features,

A key contribution in this field comes from Pérez-Rosas et al. [74], who pioneered the use of **multimodal deception detection** by combining speech, facial, and text features extracted from **courtroom trial videos**. Their results showed that combining modalities improved accuracy significantly compared to using single sources and highlighted the real-world viability of deception detection beyond lab environments.

King and Neal [49] extended this work through a comprehensive **systematic review**, aggregating and analyzing findings from dozens of deception detection studies. They emphasized that **models integrating multiple data types (e.g., audio + visual)** consistently outperformed unimodal ones. Importantly, their work brought attention to **recurrent neural architectures** like LSTM and BiGRU, which are capable of modeling long-range behavioral dependencies, a key challenge in deception analysis.

Bahaa et al. [73] advanced the state of the art by proposing a deep learning system that fuses **CNNs and LSTMs** to jointly learn from visual and vocal streams. This hybrid architecture was evaluated on benchmark datasets and outperformed earlier ML-based systems, achieving **over 90% classification accuracy**. Their work demonstrated that combining spatial feature learning with temporal modeling captures richer behavioral patterns linked to deception.

On the visual processing front, ViT (Vision Transformers) introduced by Dosovitskiy et al. [71] have emerged as powerful alternatives to CNNs. Though originally applied to general computer vision tasks, recent research has adapted ViT to analyze **micro-expressions in deception detection**, due to their ability to model long-range dependencies between image patches. Similarly, Yan et al. [70] introduced **Spatial-Temporal Graph Convolutional Networks (ST-GCN)** that work well with **skeleton-based data**, making them suitable for identifying **full-body motion inconsistencies** often linked to deceptive behavior.

Together, these contributions reflect a broader trend: systems that leverage deep architectures and fuse multiple behavioral cues tend to generalize better, achieve higher accuracy, and are more robust to noise. However, several challenges remain open. Many of the reviewed models rely on relatively small or domain-specific datasets, limiting

generalizability. Additionally, ethical considerations such as bias, fairness, data privacy, and interpretability are often under-addressed, raising concerns about the deployment of such technologies in high-stakes applications like law enforcement and security. Future research should aim to improve transparency, expand datasets, and incorporate ethical frameworks to ensure responsible use of AI in deception detection

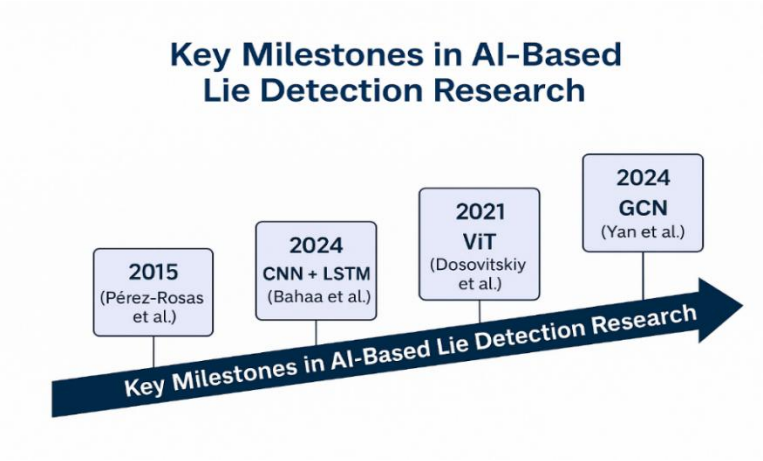


Figure 22 Key Milestones in AI-Based Lie Detection Research (2015–2024)

II.6. Datasets in Lie Detection Research

II.6.1. Importance of Datasets

High-quality datasets play a pivotal role in the development and evaluation of lie detection systems. In artificial intelligence, particularly machine learning and deep learning—models rely on data to learn behavioral patterns associated with truthful or deceptive behavior. The accuracy, generalizability, and fairness of these models are directly linked to the quality, diversity, and representativeness of the datasets used for training and testing.

Unlike traditional lie detection techniques that rely on subjective human interpretation, AI-based approaches require large volumes of annotated data to identify subtle patterns across modalities such as voice tone, facial expressions, body posture, and linguistic cues. These datasets must not only contain truthful and deceptive samples but also offer a wide range of variation across speakers, environments, emotions, and cultures to avoid overfitting and model bias [49].

Furthermore, the importance of real-life, **naturalistic deception data** cannot be overstated. Many early studies used **acted or laboratory-based deception**, which often lacks the emotional realism and unpredictability of genuine lies. As a result, models trained on such data may perform well in controlled settings but fail when applied in real-world scenarios. This challenge has led to increased interest in datasets like the **Real-Life Trial Deception Dataset (RLDD)**, introduced by Pérez-Rosas et al. [74], which captures multimodal data from actual courtroom trials, and the **Bag-of-Lies Dataset**, which records individuals telling true and false autobiographical stories in spontaneous conditions.

Multimodal datasets that include audio, video, and textual data have proven especially valuable, as they allow models to integrate different behavioral cues. Bahaa et al. [73], for example, demonstrated that combining audio spectrograms and video frames significantly improved deception detection accuracy compared to unimodal data, reinforcing the value of rich, multi-source datasets.

Beyond technical performance, the **ethical quality of datasets** is equally important. Issues of consent, privacy, and bias must be addressed in the collection and labeling of deception data. For instance, using publicly recorded trial videos requires ethical justification, and any personally identifiable information must be handled responsibly. As Sreerama and Krishnamoorthy [50] and Floridi et al. [77] note, ethical AI development hinges on fairness, transparency, and respect for data subjects' rights.

In summary, datasets are not just inputs for algorithms, they are the foundation of trustworthy AI systems. The ongoing advancement of deception detection research depends on access to **diverse, ethically sourced, and multimodal datasets** that reflect the complexity of real-world human behavior.

II.6.2. Existing Lie Detection Datasets

Over the past decade, several datasets have been developed to support the training and evaluation of AI-based deception detection systems. These datasets vary in terms of data modality, context of collection (laboratory vs. real-world), availability, and the type of deception involved (e.g., high-stakes courtroom lies versus low-stakes autobiographical narratives). The availability of diverse and representative datasets is

essential for developing models that are not only accurate but also robust to real-world conditions.

a. Real-Life Trial Deception Dataset (RLDD)

Introduced by Pérez-Rosas et al. [74], the RLDD is one of the most cited multimodal deception datasets. It includes video clips from real courtroom trials, labeled as either truthful or deceptive by a panel of annotators. The dataset captures naturalistic behavior and includes audio, visual, and text modalities. Due to the high-stakes nature of courtroom scenarios, it provides valuable insight into genuine emotional and behavioral cues.

b. Bag of Lies Dataset

The Bag-of-Lies dataset is a multimodal dataset consisting of video, audio, eye gaze, and EEG data from 35 unique subjects. Each subject was shown 6–10 images and asked to describe them truthfully or deceptively. The dataset includes 325 manually annotated recordings (162 lies and 163 truths). While it is sometimes referenced as publicly accessible, it is in fact protected with a password and requires a formal request to access. [78].

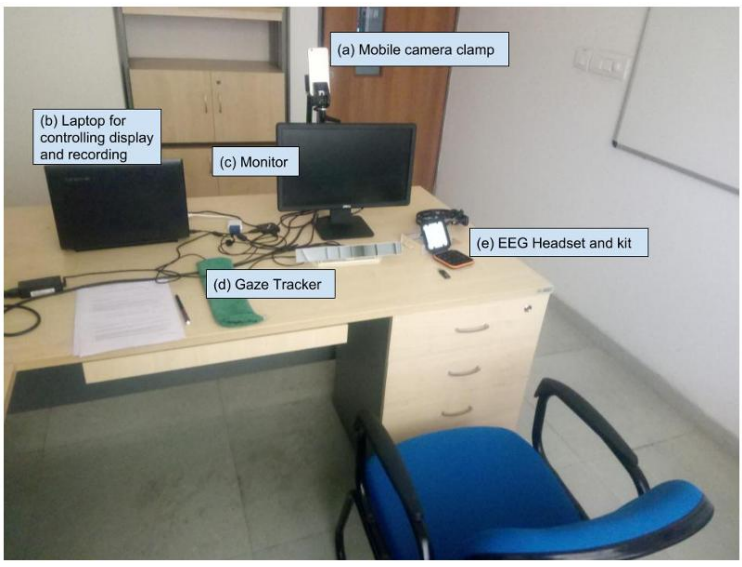


Figure 23 Experimental setup used in the Bag-of-Lies dataset [78].

c. Silesian Deception Dataset (SDD)

The Silesian Deception Dataset comprises 101 video recordings of subjects responding to questions designed to elicit truthful or deceptive answers. The recordings were captured

at 100 frames per second using a high-speed camera. The dataset includes annotations for various facial expressions and movements, making it suitable for analyzing nonverbal cues associated with deception [79].

d. Celeb-Lie Dataset

Celeb-Lie contains **interview clips** of public figures answering controversial or emotionally charged questions. It is labeled by experts and supplemented with crowd-sourced annotations [80]. Although not publicly available due to licensing restrictions, Celeb-Lie is notable for its use in evaluating models under **unpredictable and emotionally intense conditions**.

e. Columbia Deception Corpus (CDC)

The Columbia X-Cultural Deception Corpus contains within-subject deceptive and non-deceptive speech from native speakers of Standard American English. The dataset includes audio recordings of participants responding to questions in both truthful and deceptive manners [81]. It is designed to support research in cross-cultural deception detection and analysis of speech patterns.

f. Multimodal Deceptive Personal Expression (MDPE) Dataset

The MDPE dataset [82] is a large-scale multimodal deception dataset that includes audio, video, text, and eye-tracking recordings from 193 participants responding either truthfully or deceptively. It is not publicly available and requires a request and approval process to obtain access

Table 4 Comparison of Common Lie Detection Datasets.

Dataset	Modality	Environment	Type of Deception	Public?	Notes
RLDD [21]	Audio, Video, Text	Real World (Courtroom)	High-stakes, natural	✅	Multimodal; used in many DL studies
Bag of Lies [95]	Audio, Video	Semi-controlled	Low-stakes, autobiographical	❌	Good for spontaneous deception analysis
Silesian Dataset [96]	Audio, Video, Physiological	Lab-controlled	Low-stakes, instructed	✅	Ideal for clean sensor fusion studies
Celeb-Lie [97]	Audio, Video	Media/interviews	Real-world, emotional	❌	Expert-labeled; limited access due to licensing
Columbia (CDC) [98]	Audio, Video, Eye, Facial	Lab-controlled	Induced deception	✅	Includes micro-expression and gaze data
MDPE [82]	Audio, Video, Text, Eye Tracking, Emotion	Semi-controlled Interview	Prompted deception, emotional traits	❌	Request required; only audio used in this study

II.6.2.1. Dataset Used in This Study

This study makes use of the Real-Life Trial Deception Dataset (RLDD) [21], developed by researchers at the University of Michigan. The dataset comprises 121 short video clips, each labeled as either truthful or deceptive, based on courtroom testimonies. Each clip contains synchronized audio and video, allowing for multimodal analysis of speech, facial micro-expressions, and head movements in real-world, high-stakes environments.

Figure 24 Sample frames from the Real-Life Trial Deception Dataset (RLDD) [86].

Only the **audio and visual modalities** were used in this work; the text transcripts and gesture annotations were excluded. Audio features were extracted using the **OpenSMILE toolkit**, while video frames were processed using the **Vision Transformer (ViT)** architecture. The dataset was balanced to contain **60 truthful and 60 deceptive** samples, and was split into training, validation, and testing sets using a **70/10/20 ratio**.

The RLDD is well-suited for multimodal deception detection, as it provides **realistic behavioral data** under legal scrutiny, making it a strong foundation for evaluating the performance of both machine learning and deep learning models. However, the limited sample size and potential demographic imbalances represent challenges that are addressed in later sections.

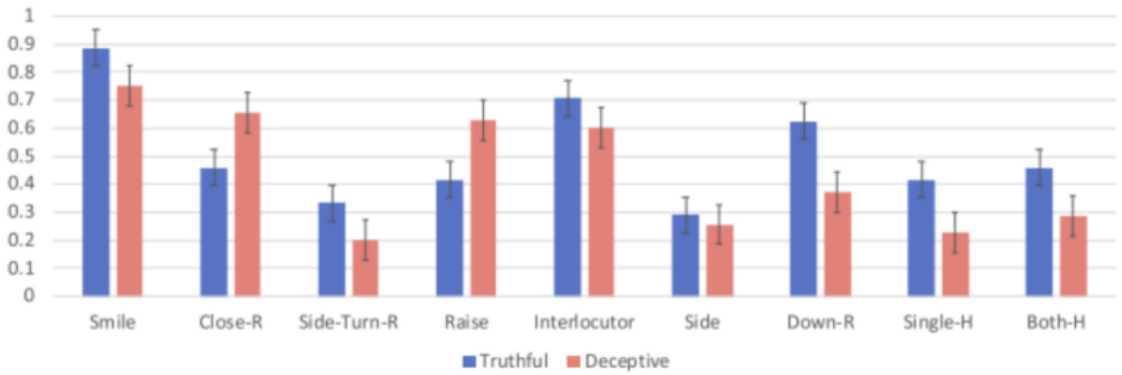


Figure 25 Comparison of Gesture and Expression Distributions [86].

II.6.3. Dataset Challenges and Ethical Considerations

Despite the progress made in collecting and using datasets for deception detection, several challenges persist that directly affect the performance, fairness, and applicability of AI-based models.

a. Limited Dataset Size

One of the most pressing issues is the small size of publicly available deception datasets. For example, the Real-Life Trial Deception Dataset (RLDD) contains only 121 videos. Such limited datasets make it difficult to train large-scale deep learning models without overfitting, and they limit generalization to new contexts or populations.

b. Lack of Diversity and Representativeness

Most datasets feature subjects from restricted demographic groups—in terms of gender, ethnicity, age, or cultural background—which introduces a risk of bias. Models trained on such skewed data may not perform reliably across diverse populations. In deception detection, this is particularly sensitive, as misclassification can have serious ethical and legal consequences.

c. Labeling Subjectivity

Deception labeling often relies on human annotators or secondary cues (e.g., trial outcomes), which can introduce subjectivity and inconsistency. Unlike tasks like object detection, there is no universally observable "ground truth" for deception, making annotation quality a key concern.

d. Modality Limitations

Some datasets offer only one or two modalities (e.g., audio without visual cues), which restricts their usefulness for multimodal learning. Others require extensive manual preprocessing (e.g., face alignment, gesture labeling), limiting scalability.

e. Ethical and Legal Concerns

Datasets like RLDD use public trial recordings, which—although legally accessible—raise questions around consent, privacy, and dignity of the individuals involved. Even if anonymized, repurposing courtroom footage for AI training may trigger ethical debates, especially in high-stakes domains like criminal justice.

As Floridi et al. [77] and Sreerama & Krishnamoorthy [50] argue, responsible AI development must include fairness, explainability, and informed consent as core principles. Deception detection systems trained on ethically questionable or biased data risk reinforcing harmful outcomes in areas such as hiring, border control, or law enforcement.

II.7. Comparative Analysis: Traditional vs. AI-Based Deception Detection Methods

Over the years, deception detection has evolved from physiological signal monitoring and human judgment to advanced artificial intelligence (AI) systems that leverage audio-visual and linguistic cues. While traditional methods like polygraph tests and facial analysis have been in use for decades, AI-based approaches offer new capabilities that are reshaping the field. This section provides a structured comparison between both paradigms, highlighting their strengths, limitations, and practical implications.

II.7.1. Comparison Criteria and Metrics

The main criteria for evaluating deception detection methods include:

- **Accuracy**: The proportion of correct predictions (truthful or deceptive)
- **Objectivity**: The method's resistance to human bias
- **Scalability**: Ability to deploy across settings with minimal expert intervention
- **Interpretability**: How understandable the method's decision-making is
- **Intrusiveness**: Whether the technique invades privacy or causes discomfort
- **Real-Time Use**: Suitability for dynamic or live applications
- **Ethical Risk**: Potential for bias, misuse, or harm

II.7.2. Comparative Discussion and Analysis

Traditional lie detection methods such as the polygraph, facial expression coding, and voice stress analysis (VSA) are rooted in biological and psychological theories. While

these techniques have historical significance, they remain subject to **high variability**, **human interpretation**, and are often **non-scalable**. For instance, polygraph results can be influenced by anxiety unrelated to lying, and FACS coding requires extensive training and time, limiting its practical use.

In contrast, AI-based methods such as machine learning classifiers and deep neural networks can **analyze complex behavioral patterns** from large datasets, identify deception with **higher consistency**, and scale easily through automation. For example, recent studies using convolutional and transformer-based models have reported accuracies exceeding 90% in multimodal deception detection tasks [73].

However, the advantage of automation comes with caveats. While traditional methods are often **legally accepted and interpretable**, AI methods face criticism for being "black boxes," lacking transparency in how predictions are made. Furthermore, their effectiveness depends heavily on **dataset quality**. As discussed in Section 2.5.3, limited or biased data can lead to **algorithmic discrimination**.

To summarize the comparison, the following table outlines the strengths and limitations of each paradigm:

Table 5 Comparative Overview: Traditional vs. AI-Based Deception Detection.

Criterion	Traditional Methods	AI-Based Methods
Accuracy	Variable (60–85%)	Often higher (80–95%) with good datasets
Objectivity	Susceptible to human bias	Data-driven and less subjective
Real-time Capability	Low (manual processing)	High (can be real-time with appropriate setup)
Interpretability	Moderate to high	Often limited (depends on model architecture)
Scalability	Low	High (once trained, easily deployed)
Intrusiveness	Often intrusive (e.g., sensors, wires)	Can be passive (e.g., video/audio analysis)

Criterion	Traditional Methods	AI-Based Methods
Ethical Considerations	Known, but manageable (e.g., polygraph protocols)	Emerging concerns (e.g., consent, algorithmic bias)

While both paradigms have roles to play depending on context, AI-based deception detection offers **significant promise** in terms of automation, objectivity, and performance — provided that ethical and data challenges are responsibly addressed. Integrating the **strengths of both approaches** — such as combining AI-driven alerts with expert human judgment — may represent the most balanced path forward.

II.8. Conclusion

Deception detection has transitioned from subjective, human-led evaluations to more objective, algorithmic systems. Traditional techniques, while still in use, often lack consistency and generalizability. In contrast, AI-based methods offer scalability and greater precision by learning deception-related patterns from data. This chapter reviewed key models, datasets, and evaluation strategies that inform current research in the field, setting the stage for experimental implementation in the next chapter.

Chapter III. Methods and Experiments

III.1. Introduction

This chapter presents the full methodology adopted to design, implement, and evaluate a multimodal deception detection system. The approach integrates audio, visual, and text-based modalities to identify deceptive behavior using both classical machine learning and advanced deep learning techniques. The chapter begins by detailing the experimental dataset (RLDD) and the preprocessing procedures used to standardize and extract features from raw audio-visual-text data. Each modality is then transformed into fixed-length feature representations using domain-relevant techniques such as ComParE_2016, MFCC, Vision Transformers, Whisper, and BERT embeddings. Fusion strategies—both early and late—are employed to combine information across modalities. A wide range of models are implemented and tested under unified training and evaluation conditions, and all experiments are supported by a comprehensive software stack that ensures reproducibility and scalability. This experimental design aims to fairly assess the strengths and limitations of each method across different configurations.

III.2. Methods

This section presents the modeling strategies employed in the proposed deception detection system. The aim is to evaluate the ability of different machine learning and deep learning algorithms to classify statements as truthful or deceptive based on multimodal input. Each model was tested under three experimental conditions: using audio-only features, visual-only features, and combined representations through fusion techniques.

Two fusion strategies were adopted. In **early fusion**, feature vectors from different modalities are combined before being passed to a single model. In **late fusion**, separate models are trained independently on each modality, and their predictions are combined at the decision level. This comparative framework ensures a fair assessment of each algorithm's strengths and limitations across different input configurations.

III.2.1. Machine Learning

Classical machine learning algorithms were used to establish interpretable and efficient baselines for deception detection. These models were trained to perform binary classification on feature vectors derived from individual or fused modalities. The machine

learning models tested include **Support Vector Machine**, **Logistic Regression, Decision Tree**, **Random Forest**, and **XGBoost**.

All models were subjected to hyperparameter tuning using grid search, and their performance was evaluated using standard classification metrics such as accuracy, precision, recall, and F1-score.

III.2.2. Deep Learning

Deep learning models were used to capture complex, nonlinear patterns that may not be easily handled by classical approaches. These models are especially suited for learning from sequences (e.g., speech or frame data) and high-dimensional embeddings.

The deep learning architectures tested include **Conv1D**, bidirectional recurrent models (**BiGRU**, **BiLSTM**), **CNN+LSTM**, **ViT**, and **GCN**. Each architecture was evaluated under the same input conditions (audio-only, visual-only, early fusion, and late fusion), and optimized using grid search. Evaluation metrics included classification accuracy and training/validation stability.

III.3. Experimental Design

III.3.1. Dataset

In this study, we used the **Real-Life Trial Deception Dataset (RLDD)**, a publicly available dataset specifically designed for multimodal deception detection tasks. RLDD contains video recordings of real courtroom testimonies, each labeled as either **truthful** or **deceptive** based on judicial outcomes and supporting evidence. The dataset includes synchronized audio and visual streams, which allows for a comprehensive analysis of both verbal and non-verbal behavioral cues.

In total, the dataset consists of **121 samples**, with each sample being a short video clip focused on a single testimony. Each clip is annotated with a **binary label**: **1** for *lie* and **0** for *truth*. This setup presents a realistic and challenging benchmark due to the subtlety of deceptive behavior in natural settings.

RLDD was selected for its authenticity, natural variation in speaker behavior, and the presence of aligned audio-visual data — all of which are essential for building and evaluating effective deception detection models.

Figure 26 illustrates the structure of the RLDD dataset used in this study. The raw courtroom recordings were segmented into individual short clips, each containing a single testimony. For each clip, both audio and video streams were extracted, and binary labels were assigned to indicate whether the testimony was truthful (0) or deceptive (1).

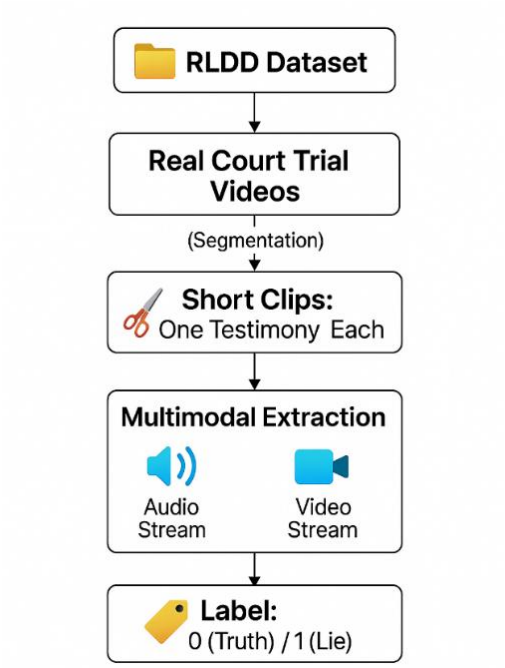


Figure 26 RLDD Dataset Preprocessing Pipeline.

III.3.2. Data Preprocessing

Before extracting features or training models, several preprocessing steps were applied to standardize the input data. For audio data, all video files in **.mp4** format were processed using MoviePy to extract the audio track in **.wav** format. These audio files were normalized and trimmed to remove silence or noise. For visual data, each video was sampled at 1 frame per second using **FFmpeg**. The resulting frames were resized to 224×224 pixels and normalized to match the expected input of the Vision Transformer model. For text, Whisper was used to transcribe the spoken content from audio clips into plain text, and basic cleaning steps were applied to remove non-verbal artifacts such as

"[laugh]" or "[noise]". All data were saved in **.npy** or **.csv** format to enable fast loading during experiments.

III.3.3. Feature Extraction

To enable automated deception detection from raw multimedia inputs, each modality—audio, visual, and text—was processed to extract meaningful, fixed-length feature vectors suitable for classification. The objective of this stage is to convert complex, high-dimensional input data into structured representations that retain the relevant behavioral and linguistic cues. The following subsections describe the feature extraction techniques applied to each modality used in this study.

III.3.3.1. Audio Feature Extraction

To analyze speech-based cues related to deception, two types of acoustic features were extracted from each audio clip in the RLDD dataset: the **ComParE_2016 feature set** and **Mel-Frequency Cepstral Coefficients (MFCCs)**. These two methods capture different aspects of the speech signal—ComParE is focused on paralinguistic and emotional patterns, while MFCCs reflect the spectral shape of speech in a way that mimics human hearing.

a. ComParE_2016 Feature Set

The ComParE_2016 (Computational Paralinguistics Challenge) feature set was extracted using the `ComParE_2016.conf` configuration in the OpenSMILE toolkit. This configuration computes over 60 Low-Level Descriptors (LLDs), including prosodic (e.g., energy, pitch), spectral (e.g., MFCCs, spectral flux), and voice quality features (e.g., jitter, shimmer).

For each LLD *x(t)*, a large set of **functionals** *f(x)* such as mean, standard deviation, percentiles, skewness, kurtosis, and regression coefficients—are applied over the full audio segment or over time windows:

$$f(x) = \{\mu, \sigma, \text{percentiles}, \text{slope}, \text{zero} - \text{crossings}, \dots\} \quad \text{.....EQ 16}$$

These functionals transform the time-series LLDs into fixed-length feature vectors. The final output is a **6373-dimensional vector** that represents the full audio clip.

This feature set captures expressive patterns such as vocal tension, pitch variation, and articulation dynamics, which are important for detecting deception [83].

b. MFCC (Mel-Frequency Cepstral Coefficients)

MFCCs were extracted using the Librosa library in Python. The process starts by dividing the audio signal into overlapping frames, then calculating the **Short-Time Fourier Transform (STFT)** to obtain the power spectrum $|\,X(f)\,|^2$. This is passed through a set of Mel-scaled filterbanks $H_m(f)$, followed by a **logarithmic transformation** and a **Discrete Cosine Transform (DCT)** to produce decor-related cepstral coefficients:

$$MFCC_K = \sum_{m=1}^{M} \log\left(|\,X(f)\,|^2\, H_m(f)\right) .\cos\left[\frac{\pi k(m-0.5)}{M}\right] \text{.....EQ 17}$$

We extracted:

- 13 static MFCCs
- 13 delta (first derivative)
- 13 delta-delta (second derivative)
- 3 RMS-based energy features (RMS, delta RMS, delta-delta RMS)

These components were concatenated into a **42-dimensional feature vector**, then averaged across frames to create one vector per clip.

MFCCs are widely used in speech recognition but may not capture emotional subtleties as effectively as ComParE_2016 [84].

Figure 27 illustrates the audio feature extraction pipeline applied to the RLDD dataset, detailing the conversion of video recordings into `.wav` files and the subsequent extraction of ComParE_2016 and MFCC features using OpenSMILE and Librosa, respectively.

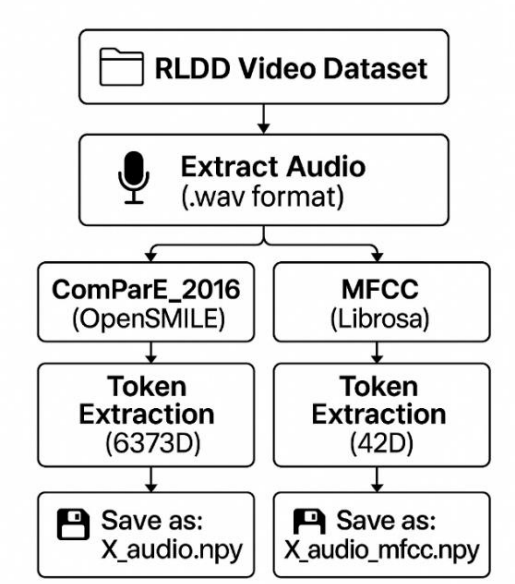


Figure 27 Audio feature extraction process from the RLDD dataset.

III.3.3.2. Visual Feature Extraction – ViT Embeddings

To extract visual features from the Real-Life Trial Dataset (RLDD), a Vision Transformer (ViT)-based pipeline was implemented. The process involved the following stages:

First, each video was sampled uniformly at a rate of **1 frame per second (fps)** using **FFmpeg**, ensuring a consistent temporal representation across all clips regardless of their duration. This sampling strategy balances the need for temporal coverage with computational efficiency.

Next, the extracted frames were **resized to 224×224 pixels** and normalized to match the input format expected by standard ViT models. A pre-trained `google/vit-base-patch16-224` model from Hugging Face Transformers was then applied. For each input frame, the ViT generated high-dimensional feature representations. Specifically, the **[CLS] token** from the final hidden state was extracted as a **768-dimensional vector**, summarizing the global context of the frame.

To represent an entire video, the 768-D embeddings from all frames were aggregated using the **mean pooling** technique. This step resulted in a single vector per video sample with shape **(768,)**, which captures overall visual characteristics relevant to deception.

Finally, the visual embeddings were stored in a .npy file format (X_visual.npy) and later used as input to both classical and deep learning models. This approach allowed the system to leverage powerful transformer-based visual representations while maintaining computational tractability.

Figure 28 illustrates the visual feature extraction pipeline. Frames were sampled from RLDD videos at a rate of 1 frame per second using FFmpeg, resized and normalized, and passed through a pre-trained Vision Transformer (ViT). The `[CLS]` token embeddings were averaged to produce a single 768-dimensional vector per video.

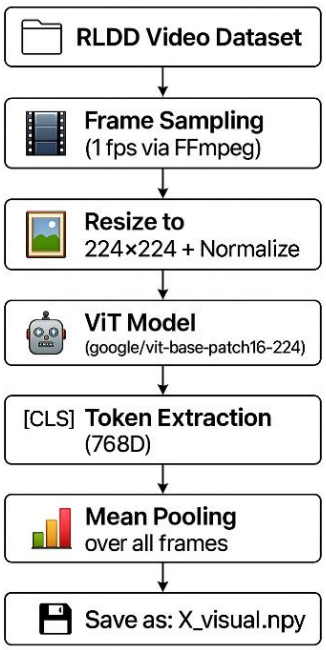


Figure 28 Visual feature extraction process from the RLDD dataset.

III.3.3.3. Text Features

To explore the potential of verbal cues in deception detection, automatic speech transcriptions were generated using the Whisper model. These transcriptions were then processed using a pretrained BERT model from Hugging Face Transformers, which embedded each sentence into a fixed-length 768-dimensional semantic vector. The resulting text embeddings were used in initial classification experiments to assess the discriminative value of linguistic information. While promising, the text modality showed limited performance in comparison to audio and visual cues in the final evaluation.

Figure 29 shows the text-based feature extraction process. Audio files were transcribed using the Whisper speech-to-text model. The resulting transcripts were embedded using a fine-tuned BERT model for classification into "truth" or "lie."

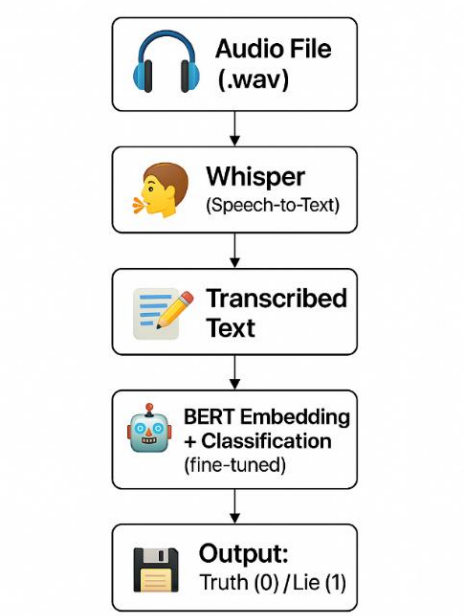


Figure 29 Text-Based Deception Detection Pipeline.

III.3.4. Fusion Strategies

In multimodal deception detection, fusion strategies are critical for leveraging complementary information from different modalities—in this case, audio and visual cues. This section presents and compares two widely adopted fusion approaches: early fusion and late fusion.

III.3.4.1. Early Fusion

Early fusion refers to the integration of features from multiple modalities at the input level. Specifically, the audio feature vector (e.g., ComParE_2016 or MFCC) and the visual embedding (e.g., ViT-based 768D vector) are concatenated into a single high-dimensional feature vector. This fused vector is then used as input to a unified model, allowing the learning algorithm to discover joint patterns across modalities during training.

This strategy has the advantage of enabling the model to learn complex cross-modal interactions. However, it also introduces certain challenges such as:

- Feature imbalance (e.g., 6373D audio vs. 768D visual)
- Increased dimensionality
- Risk of overfitting on small datasets

Despite these limitations, early fusion is straightforward to implement and often performs well when modalities are cleanly aligned and complementary.

Figure 30 illustrates the early fusion strategy adopted in this study. Audio and visual features are first standardized and temporally aligned, then concatenated into a single multimodal vector, which is used to train a unified classification model.

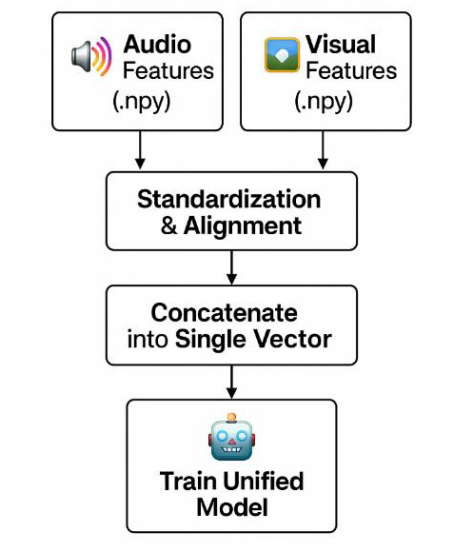


Figure 30 Early Fusion Architecture.

III.3.4.2. Late Fusion

Late fusion adopts a decision-level integration approach. In this setup, separate models are trained independently on each modality—one on audio features and the other on visual features. These models may either share the same architecture or use different architectures optimized for each modality.

During inference, each model generates prediction probabilities for its respective modality. The final decision is obtained by **averaging** these probabilities using **soft voting**.

This strategy offers several advantages:

- Each model can specialize in processing its own modality
- It allows flexible use of different architectures per modality
- It enhances robustness when one modality is noisier or underperforms

This method was applied consistently across multiple model combinations to evaluate the impact of decision-level fusion on deception detection accuracy.

Figure 31 depicts the late fusion strategy used in this study. Predictions from separately trained audio and visual models are combined using soft voting, resulting in a final binary classification of truth or lie.

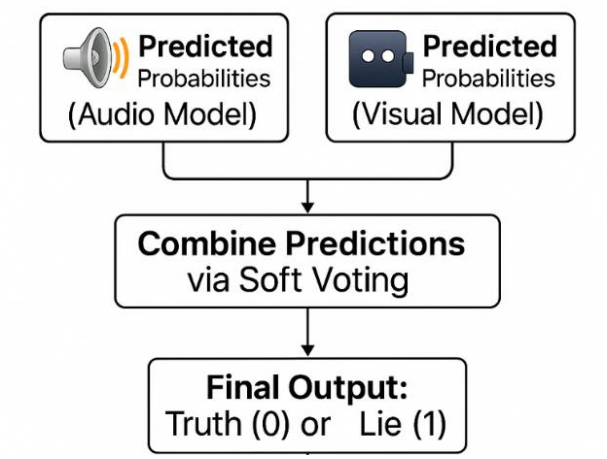


Figure 31 Soft Voting Fusion Architecture.

III.3.5. Model Training Strategy

To ensure consistency and fairness across all experiments, a unified training strategy was applied to both machine learning and deep learning models. Most models were trained using an **80/20 train-test split**, where 80% of the RLDD samples were allocated for training and 20% for evaluation. This ratio was chosen due to the relatively small size of the dataset (121 samples), as it offered a more stable test set compared to alternatives such as 95/5 or 75/25, which led to overly small evaluation subsets and unreliable metrics. In select deep learning experiments, **5-Fold Cross-Validation** was also employed to enhance model generalization. The dataset was divided into five equal folds, and each fold was used once as a validation set while the remaining served as training data. Final results were averaged across the five runs to reduce performance variance.

For hyperparameter optimization, **Grid Search** was used across all models to identify the best configurations. In the case of classical machine learning algorithms such as SVM, XGBoost, Random Forest, Logistic Regression, and Decision Tree, grid search was performed on key parameters including kernel type, penalty, regularization constants (C, gamma), number of estimators, and maximum tree depth. For deep learning models—Conv1D, BiLSTM, BiGRU, CNN+LSTM, ViT, and GCN—the tuned parameters included the number of convolutional filters or GRU/LSTM units, dropout rates, activation functions (e.g., ReLU or GELU), and transformer-specific components such as head size and encoder depth. All experiments were repeated under multiple input configurations, including audio-only, visual-only, early fusion, and late fusion, ensuring a balanced and exhaustive evaluation of each model's behavior.

III.3.6. Model Evaluation

To evaluate model performance, a range of standard classification metrics was used. These include:

- **Accuracy**, which reflects the proportion of correctly predicted samples.
- **Precision**, indicating how many predicted positives were correct.
- **Recall**, measuring how many actual positives were identified.
- **F1-score**, the harmonic mean of precision and recall.
- **Confusion matrices**, providing insight into the distribution of correct and incorrect predictions.
- **Training and validation curves**, used in deep learning models to observe learning behavior and detect overfitting.

Note: Due to the test set containing only 25 samples (20% of 121), accuracy values were often discrete (e.g., 92.00%, 96.00%), as each misclassified instance caused a 4% change. These metrics ensured consistent and interpretable comparisons across modalities and fusion strategies.

III.3.7. Software and Computational Resources

All experiments in this study were conducted using the **free version of Google Colaboratory (Colab)**, a cloud-based Jupyter notebook platform that supports GPU acceleration. The project was developed entirely in **Python 3.10**, which offers extensive support for machine learning, deep learning, and data processing. This section outlines all tools, libraries, and frameworks used for feature extraction, preprocessing, model development, evaluation, and visualization.

a. Development Environment

- **Platform**: Google Colab (Free version)
- **Operating System (runtime)**: Ubuntu 20.04 (Colab backend)
- **Programming Language**: Python 3.10

b. Core Utilities and Data Handling

- **numpy, pandas:** Numerical and tabular data manipulation
- **random, os, glob:** For reproducibility, path handling, and automated file loading
- **joblib:** For saving and reloading trained models (ML)

c. Machine Learning Libraries

- **scikit-learn:**
  - Models: SVM, Logistic Regression, Decision Tree, Random Forest
  - Tools: train_test_split, StandardScaler, GridSearchCV, classification_report, confusion_matrix
- **xgboost:** For gradient boosting classifier implementation
- **scikit-learn.metrics**: For computing accuracy, precision, recall, and F1-score
- **lime:** For local interpretability of ML and DL models (e.g., SVM, BiGRU)

d. Deep Learning Frameworks

- **tensorflow, keras:**

- Used for Conv1D, BiLSTM, BiGRU, CNN+LSTM, ViT implementations
- Components: Sequential, Model, Dense, Conv1D, Dropout, LSTM, GRU, EarlyStopping, etc.

- **torch (PyTorch):** Required for Whisper and Hugging Face Transformers
- **spektral:** For Graph Convolutional Networks (GCN)
- **torch_geometric:** Additional support for GCN (if used during implementation)

e. Audio Feature Extraction

- **OpenSMILE**:
  - Used via command-line with ComParE_2016.conf to extract standardized paralinguistic features
- **librosa:**
  - Used for MFCC feature extraction including static, delta, delta-delta, and RMS features
- **scipy.io.wavfile:** For loading .wav audio files when needed

f. Video and Visual Feature Extraction

- **ffmpeg:** For frame extraction from .mp4 video files (1 frame per second)
- **moviepy.editor:** To convert video to audio (.wav)
- **PIL.Image, cv2 (OpenCV):** For resizing and normalizing image frames
- **transformers (Hugging Face):**
  - Used to load the pre-trained ViT model (google/vit-base-patch16-224) for visual embeddings

g. Text Processing and Embedding

- **whisper (by OpenAI):** Used to transcribe speech to text from audio
- **transformers:**

- Used to load and fine-tune the BERT model for sentence embeddings and classification

h. Evaluation and Visualization

- **matplotlib, seaborn:** For generating accuracy/loss curves, confusion matrices, and result plots
- **scikit-learn.metrics:** For generating evaluation metrics and confusion matrices
- **lime:** Used for model explanation visualizations

i. Data Storage Formats

- **.npy**: Used to store NumPy arrays of features (X_audio.npy, X_visual.npy, etc.)
- **.csv**: Used to store intermediate feature extraction outputs (e.g., MFCC values)

This comprehensive software setup ensured the smooth execution of all steps in the project — from audio/video preprocessing to model training, evaluation, and interpretability. Despite relying on the free version of Google Colab, the environment provided adequate computational support for the relatively lightweight RLDD dataset.

III.4. Conclusion

In summary, this chapter has detailed the full experimental methodology used to investigate deception detection across audio, visual, and text modalities. From dataset preparation and preprocessing to feature extraction and fusion strategies, each step was carefully structured to enable robust model evaluation. Both classical and deep learning models were implemented, optimized using grid search, and tested under consistent experimental settings. The software environment, tools, and frameworks were clearly defined to ensure transparency and reproducibility. This methodological foundation sets the stage for the next chapter, which presents and analyzes the results obtained from the experiments and highlights the key findings of the study.

Chapter IV. Results and Discussions

IV.1. Introduction

This chapter presents the experimental framework, model architectures, and evaluation results of the proposed deception detection system. It outlines the machine learning and deep learning models tested across different input modalities—audio, visual, and their fusion. Each model was implemented using standardized preprocessing and training procedures to ensure comparability. The chapter also includes a comparative performance analysis of feature types and fusion strategies, culminating in the selection of final models for deployment.

IV.2. Model Architectures and Implementation

This section outlines the core models employed in the deception detection system. Both classical machine learning and deep learning approaches were explored using modality-specific features (audio and visual) as well as multimodal fusion strategies. Each model was trained and evaluated following standardized preprocessing and experimental setups to ensure fair comparison. The subsections that follow detail the architectures, training strategies, and performance outcomes for each model family.

IV.2.1. Classical Machine Learning Models

This section presents the classical machine learning models evaluated for deception detection. All models were tested across four input modalities: **audio-only**, **visual-only**, **early fusion** (concatenated audio and visual features), and **late fusion** (soft voting of model predictions).

- **Audio features** were extracted using the **ComParE_2016** feature set via OpenSMILE.
- **Visual features** were based on **ViT embeddings** extracted from 1-fps sampled frames.
- In **early fusion**, the feature vectors from both modalities were concatenated into a single input vector.
- In **late fusion**, models were trained separately on each modality, and final predictions were computed by averaging their probabilities (soft voting).

For each model, the best-performing hyperparameters (obtained via grid search) are listed below. Detailed performance metrics for all configurations are provided later.

IV.2.1.1. Support Vector Machine (SVM)

- **Audio (ComParE_2016):** C = 10, gamma = 'scale'
- **Visual (ViT):** C = 10, gamma = 'scale', kernel = 'rbf'
- **Early Fusion**: C = 10, gamma = 'scale'
- **Late Fusion:** Combined predictions from audio and visual SVM models using soft averaging.

IV.2.1.2. Logistic Regression

- **Audio (ComParE_2016):** C = 10, penalty = 'l2', solver = 'liblinear'
- **Visual (ViT):** C = 10, penalty = 12, solver = 'liblinear'
- **Early Fusion:** Same as audio settings
- **Late Fusion:** Soft voting between audio and visual models.

IV.2.1.3. XGBoost

- **Audio (ComParE_2016):** n_estimators = 10, max_depth = 5, learning_rate = 0.1
- **Visual (ViT):** Same as audio
- **Early Fusion:** n_estimators = 10, max_depth = 3, learning_rate = 0.1
- **Late Fusion:** Averaged predictions from audio and visual XGBoost classifiers.

IV.2.1.4. Random Forest

- **Audio (ComParE_2016):** n_estimators = 100, max_depth = 5
- **Visual (ViT):** Same as audio
- **Early Fusion:** Same as audio
- **Late Fusion:** Soft voting between audio and visual models.

IV.2.1.5. Decision Tree

- **Audio (ComParE_2016):** max_depth = 5, min_samples_leaf = 1, min_samples_split = 5
- **Visual (ViT):** max_depth = 5, min_samples_split = 2, criterion = 'entropy'
- **Early Fusion:** Same as visual
- **Late Fusion:** Late fusion applied using decision tree outputs from both modalities.

IV.2.2. Deep Learning Models

This section presents the deep learning models applied in the deception detection framework. Each model was trained and evaluated on four input configurations: audio-only (using **ComParE_2016** features), visual-only (using **ViT embeddings**), early fusion (concatenation of audio and visual features), and late fusion (soft voting of separate audio and visual models).

To ensure fair comparison and reproducibility, all models followed a **standardized training pipeline**. Unless stated otherwise in the individual sections, the following configuration was used:

- **Input Shapes:**
  - **Audio (ComParE_2016):** (60, 1)
  - **Visual (ViT Embeddings):** (96, 8)
  - **Early Fusion:** Concatenated shape based on audio and visual dimensions
  - **Late Fusion:** Combination of independent model outputs using soft voting
- **Training Setup:**
  - **Batch size:** 8
  - **Optimizer:** Adam
  - **Loss Function:** Binary Cross-Entropy
  - **EarlyStopping:** Patience = 10 (to prevent overfitting)

Each model was optimized using grid search to tune its key architecture-specific hyperparameters (e.g., number of filters or units, dropout rate, or transformer head size). The following subsections describe each architecture, its unique configuration, and accuracy results per modality.

IV.2.2.1. Conv1D Model

Conv1D → MaxPooling → Conv1D → MaxPooling → Flatten → Dense → Dropout → Sigmoid Output

The Conv1D model used in this study applies a sequence of two 1D convolutional layers with ReLU activation, each followed by MaxPooling to reduce dimensionality and retain key features. The output is flattened and passed through a Dense layer with 128 units, followed by a Dropout layer for regularization. A final sigmoid-activated output layer is used for binary classification.

Grid search parameters:

- Filters: 64, 128
- Dropout: 0.2, 0.3

Performance Summary:

The detailed accuracy results for Conv1D across all modalities are shown in **Table 6**.

Table 6 Conv1D Grid Search Summary.

Modality	Filters	Dropout	Accuracy
Audio	64	0.2	76.00%
	64	**0.3**	**88.00%**
	128	0.2	84.00%
	128	0.3	80.00%
Visual	**64**	**0.2**	**92.00%**
	64	0.3	80.00%
	128	0.2	84.00%
	128	0.3	80.00%
Early Fusion	64	0.2	84.00%
	64	**0.3**	**88.00%**
	128	0.2	80.00%
	128	0.3	88.00%
Late Fusion	–		**92.00%**

IV.2.2.2. Vision Transformer (ViT)

PatchEmbedding → TransformerEncoder ×4 → MLP Head → Sigmoid Output

The ViT model applied in this work is a custom implementation of the Vision Transformer, adapted for tabular inputs derived from audio, visual, and fused modalities. The model begins with a patch embedding layer, which splits the input into non-overlapping segments and projects them into a higher-dimensional space.

This is followed by a stack of 4 Transformer Encoder blocks, each composed of:

- Multi-head self-attention mechanism
- Feed-forward neural network
- Layer normalization
- Residual connections

The output of the final encoder is passed through an MLP head followed by a sigmoid activation for binary classification.

Grid search parameters:

- Head size: 32, 64
- Dropout: 0.1, 0.2, 0.25, 0.3

Performance Summary:

The grid search outcomes for ViT are presented in **Table 7**.

Table 7 ViT Grid Search Summary.

Modality	Head Size	Dropout	Accuracy
Audio	32	0.1	40.00%
	32	**0.2**	**64.00%**
	32	0.25	60.00%
	64	0.1	60.00%
	64	0.2	64.00%
	64	0.25	52.00%
Visual	32	0.2	84.00%
	64	**0.2**	**92.00%**
	32	0.3	88.00%
	64	0.3	76.00%
Early Fusion	**32**	**0.2**	**64.00%**
	64	0.2	56.00%
	32	0.3	56.00%
	64	0.3	56.00%
Late Fusion	-		**88.00**%

IV.2.2.3. CNN+LSTM Model

Conv1D → MaxPooling → LSTM → Dense → Dropout → Output

The CNN+LSTM model is designed to combine the spatial learning capabilities of convolutional layers with the sequential modeling power of LSTMs. The architecture follows the pattern:

Grid Search Parameters:

- Filters: 64, 128
- LSTM Units: 64, 128, 256
- Dropout : 0.1, 0.2, 0.25, 0.3

Performance Summary:

Accuracy results for CNN+LSTM under different configurations are summarized in **Table 8**.

Table 8 CNN+LSTM Grid Search Summary.

Modality	Filters	Units	Dropout	Accuracy
Audio	64	64	0.1	52.00%
	64	128	0.2	56.00%
	128	64	0.2	52.00%
	128	128	0.3	56.00%
	128	**256**	**0.25**	**60.00%**
Visual	**64**	**64**	**0.2**	**96.00%**
	64	128	0.3	92.00%
	128	64	0.2	88.00%
	128	64	0.3	88.00%
Early Fusion	64	64	0.2	72.00%
	64	64	0.3	60.00%
	128	**64**	**0.2**	**84.00%**
	128	128	0.3	56.00%
Late Fusion	-			**92.00%**

IV.2.2.4. BiGRU Model

BiGRU → Dropout → Dense → Output

The BiGRU model employed in this work consists of a bidirectional GRU layer followed by a dropout layer to reduce overfitting. The output is passed through a dense layer

activated by sigmoid for binary classification. This configuration allows the model to capture both past and future temporal dependencies in sequential data.

Grid search parameters:

- Units: 64, 128
- Dropout: 0.2, 0.3

Performance Summary:

Table 9 reports the performance of the BiGRU model across input types and settings.

Table 9 BIGRU Grid Search Summary.

Modality	Units	Dropout	Accuracy
Audio	64	0.2	68.00%
	64	0.3	72.00%
	128	0.2	72.00%
	128	**0.3**	**76.00%**
Visual	64	0.2	76.00%
	64	0.3	88.00%
	128	**0.2**	**96.00%**
	128	0.3	80.00%
Early Fusion	64	0.2	68.00%
	64	0.3	68.00%
	128	**0.2**	**72.00%**
	128	0.3	76.00%
Late Fusion	–		**92.00%**

IV.2.2.5. BiLSTM Model

BiLSTM → Dropout → Dense → Output

The BiLSTM model in this study utilizes a bidirectional LSTM layer, followed by a dropout layer to prevent overfitting. The bidirectional configuration enables the model to learn temporal patterns in both forward and backward directions, making it suitable for deception detection in sequential data. The final output is fed into a sigmoid-activated dense layer for binary classification.

Grid search parameters:

- Units: 64, 128
- Dropout: 0.2, 0.3

Performance Summary:

Table 10 displays the results obtained from BiLSTM under grid search.

Table 10 BiLSTM Grid Search Summary.

Modality	Units	Dropout	Accuracy
Audio	64	0.2	52.00%
	64	0.3	60.00%
	128	**0.2**	**68.00%**
	128	0.3	64.00%
Visual	**64**	**0.2**	**88.00%**
	64	0.3	88.00%
	128	0.2	88.00%
	128	0.3	84.00%
Early Fusion	64	0.2	64.00%
	64	0.3	64.00%
	128	0.2	72.00%
	128	**0.3**	**80.00%**
Late Fusion	–		88.00%

IV.2.2.6. GCN Model

GCN → BatchNorm → GCN → BatchNorm → Dropout → Dense (Sigmoid)

The Graph Convolutional Network (GCN) was designed to model relationships between sequential features through graph-structured learning. Each GCN layer operates on feature-adjacency pairs, enabling message-passing between time steps. Batch normalization was applied after each GCN layer to stabilize training, followed by a dropout layer to mitigate overfitting. The final dense layer used a sigmoid activation for binary deception classification. Unlike traditional GCNs using ReLU, this implementation used **GELU** activation for smoother gradient behavior.

- Graph structure:
  - Adjacency matrices A_audio.npy and A_visual.npy used for respective modalities.
  - Early fusion uses block_diag to merge both adjacency matrices.

Performance Summary:

Table 11 presents GCN model accuracy with varying units and dropout rates.

Table 11 GCN Grid Search Summary.

Modality	Units	Dropout	Accuracy
Audio	32	0.2	48.00%
	32	0.3	40.00%
	32	0.4	52.00%
	64	0.2	48.00%
	64	**0.3**	**56.00%**
	64	0.4	48.00%
	128	0.2	52.00%
	128	0.3	52.00%
	128	0.4	48.00%
Visual	**32**	**0.2**	**36.00%**
	32	0.3	36.00%
	32	0.4	36.00%
	64	0.2	36.00%
	64	0.3	36.00%
	64	0.4	36.00%
	128	0.2	36.00%
	128	0.3	36.00%
	128	0.4	36.00%
Early Fusion	**32**	**0.2**	**44.00%**
	32	0.3	44.00%
	32	0.4	44.00%
	64	0.2	44.00%
	64	0.3	44.00%
	64	0.4	44.00%
	128	0.2	44.00%
	128	0.3	44.00%
	128	0.4	44.00%
Late Fusion	-		**40.00%**

IV.3. Experimental Results by Modality

This section presents the classification results for each input modality: audio, visual, early fusion, and late fusion. "All models were trained using an 80/20 train-test split unless otherwise noted, with evaluation based on Accuracy, Precision, Recall, and F1-Score. Audio features were primarily extracted using the ComParE_2016 configuration via OpenSMILE, while visual features relied on ViT embeddings. These consistent inputs and evaluation strategies ensure fair comparison across models."

IV.3.1. Audio-Based Results

Table **12** presents the classification performance of classical machine learning models applied to audio features extracted using the ComParE_2016 set.

Table 12 Classification performance of machine learning models using audio features

Model	Class	Precision	Recall	F1-Score	Accuracy	Best Parameters
SVM	Deception	85.71%	92.31%	88.89%	**88.00%**	C = 10, gamma = 'scale'
	Truthful	90.91%	83.33%	86.96%		
Random Forest	Deception	90.00%	69.23%	78.26%	**80.00%**	n_estimators = 100, max_depth = 5
	Truthful	73.33%	91.67%	81.48%		
Logistic Regression	Deception	73.33%	84.62%	78.57%	**76.00%**	C = 10, penalty = 'l2', solver = 'liblinear'
	Truthful	80.00%	66.67%	72.73%		
XGBoost	Deception	72.73%	61.54%	66.67%	**68.00%**	n_estimators = 10, max_depth = 5, learning_rate = 0.1
	Truthful	64.29%	75.00%	69.23%		
Decision Tree	Deception	61.11%	84.62%	70.97%	**64.00%**	max_depth = 5, min_samples_leaf = 1, min_samples_split = 5
	Truthful	71.43%	41.67%	52.63%		

Table 13 summarizes the results obtained from deep learning models trained on the same audio features, highlighting the impact of architecture and hyperparameter tuning on deception detection accuracy.

Table 13 Classification performance of deep learning models using audio features

Model	Class	Precision	Recall	F1-Score	Accuracy	Best Parameters
Conv1D	Deception	85.71%	92.31%	88.89%	**88.00%**	filters=64, dropout=0.3
	Truthful	90.91%	83.33%	86.96%		
BiGRU	Deception	76.92%	76.92%	76.92%	**76.00%**	units=128, dropout=0.3
	Truthful	75.00%	75.00%	75.00%		
BiLSTM	Deception	72.73%	61.54%	66.67%	**68.00%**	units=128, dropout=0.2
	Truthful	64.29%	75.00%	69.23%		
ViT	Deception	60.00%	92.31%	72.73%	**64.00%**	head_size=32, dropout=0.2
	Truthful	80.00%	33.33%	47.06%		

Model	Class	Precision	Recall	F1-Score	Accuracy	Best Parameters
CNN+LSTM	Deception	58.82%	76.92%	66.67%	**60.00%**	filters=128, units=256, dropout=0.25
	Truthful	62.50%	41.67%	50.00%		
GCN	Deception	55.56%	76.92%	46.52%	**56.00%**	Units=64, dropout=0.3
	Truthful	57.14%	33.33%	42.11%		

To complement the numerical results presented in the previous tables, **Figure 32** illustrates the accuracy of all tested models on audio-only features. Among classical machine learning models, SVM achieved the highest performance with 88% accuracy. In the deep learning group, Conv1D matched this score, outperforming more complex architectures such as BiGRU and BiLSTM. These results confirm the strength of lightweight convolutional models in processing low-level acoustic features, while also highlighting the consistent effectiveness of SVM in high-dimensional settings like ComParE_2016. In contrast, GCN showed the weakest performance, likely due to its sensitivity to the structure and size of the input data.

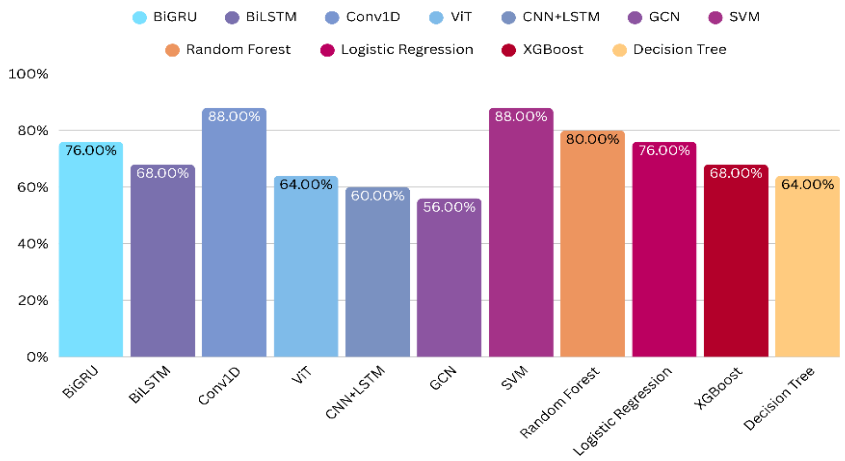


Figure 32 Accuracy of ML and DL models on audio-only features.

IV.3.2. Visual-Based Results

Table 14 displays the performance of classical machine learning models trained on Vision Transformer (ViT) embeddings derived from the RLDD video frames.

Table 14 Classification performance of machine learning models using visual features

Model	Class	Precision	Recall	F1-Score	Accuracy	Best Parameters
SVM	Deception	100%	76.92%	86.96%	**88.00%**	C = 10, gamma = 'scale', kernel = 'rbf'
	Truthful	80.00%	100%	88.89%		
Logistic Regression	Deception	73.33%	100%	78.57%	**88.00%**	C = 10, penalty = l2, solver = 'liblinear'
	Truthful	80.00%	66.67%	72.73%		
XGBoost	Deception	100%	76.92%	86.96%	**88.00%**	n_estimators = 10, max_depth = 3, learning_rate = 0.1
	Truthful	80.00%	100%	88.89%		
Random Forest	Deception	100%	69.23%	81.82%	**84.00%**	n_estimators = 100, max_depth = 5
	Truthful	75.00%	100%	85.71%		
Decision Tree	Deception	78.57%	84.62%	81.48%	**80.00%**	max_depth = 5, min_samples_split = 2, min_samples_split = 10, criterion = 'entropy'
	Truthful	81.82%	75.00%	78.26%		

Table 15 outlines the results of deep learning models applied to ViT-based visual features, demonstrating their effectiveness in capturing nonverbal deception cues.

Table 15 Classification performance of deep learning models using visual features

Model	Class	Precision	Recall	F1-Score	Accuracy	Best Parameters
CNN+LSTM	Deception	100%	92.31%	96.00%	**96.00%**	filters=64, units=64
	Truthful	92.31%	100%	96.00%		
BiGRU	Deception	92.86%	100%	96.30%	**96.00%**	units=128, dropout=0.2
	Truthful	100%	91.67%	95.65%		
ViT	Deception	86.67%	100%	92.86%	**92.00%**	head_size=64, dropout=0.2
	Truthful	100%	83.33%	90.91%		
Conv1D	Deception	92.31%	92.31%	92.31%	**92.00%**	filters=64, dropout=0.2
	Truthful	91.67%	91.67%	91.67%		
BiLSTM	Deception	85.71%	92.31%	92.31%	**88.00%**	units=64, dropout=0.2
	Truthful	90.91%	83.33%	86.96%		
GCN	Deception	38.46%	38.46%	38.46%	**36.00%**	Units=32, dropout=0.2
	Truthful	33.33%	33.33%	33.33%		

To support the tabulated results, **Figure 33** presents the accuracy scores for all models evaluated on visual features. Deep learning models significantly outperformed classical ones, with both CNN+LSTM and BiGRU achieving the highest accuracy of 96%. These results confirm the effectiveness of temporal and spatial modeling in ViT-based visual embeddings. Classical models such as SVM, XGBoost, and Logistic Regression showed competitive performance (88%), while GCN underperformed due to its sensitivity to graph-based structure and lack of sequential optimization. Overall, deep architectures proved more capable of capturing deception-related visual cues.

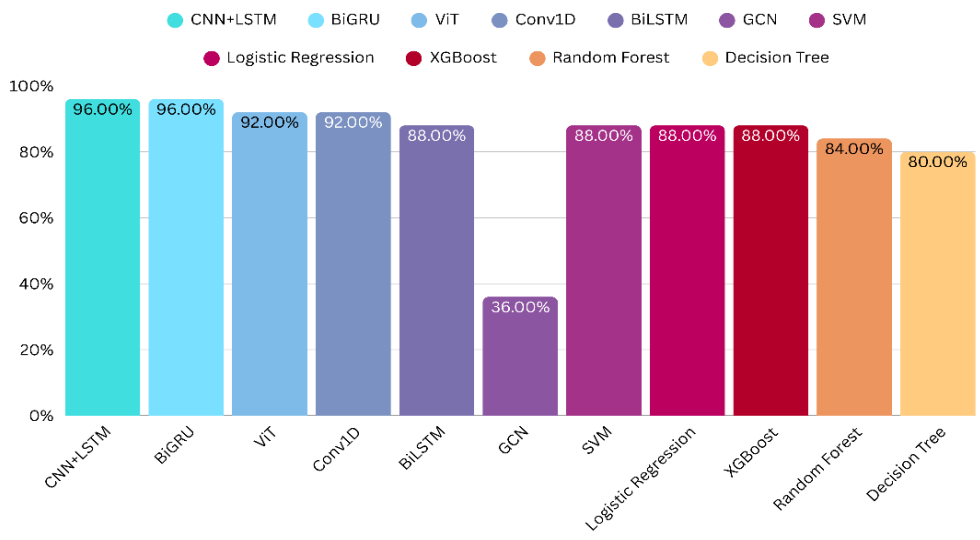


Figure 33 Accuracy of ML and DL models on visual-only features.

IV.3.3. Text-Based Results

To evaluate the impact of textual content on deception detection, transcripts were first generated from the RLD audio samples using Whisper. These were then embedded and classified using a fine-tuned bert-base-uncased model from Hugging Face. The model was trained for 4 epochs with standard BERT hyperparameters, and evaluated using a train/test split.

Despite the use of a robust transformer-based architecture, the resulting performance was limited. The model achieved an overall accuracy of 68%, indicating relatively weak predictive power when using verbal content alone.

This outcome supports the hypothesis that textual information is insufficient for detecting deception in this context. Deception is often conveyed more clearly through paralinguistic

features such as tone, hesitation, or vocal stress, rather than the specific words spoken. As a result, text-based features were not included in subsequent fusion experiments.

Table 16 Performance of BERT Text Classifier on Transcribed RLD Dataset

Metric	Value	
Accuracy	68.00%	
Precision	Deception	Truthful
	72.73%	64.29%
Recall	Deception	Truthful
	61.54%	75.00%
F1-Score	Deception	Truthful
	66.67%	69.23%

IV.3.4. Fusion-Based Results

IV.3.4.1. Early Fusion Results

Table 17 reports the classification outcomes for both classical and deep learning models using early fusion, where audio (ComParE_2016) and visual (ViT) features were concatenated before training.

Table 17 Performance of ML and DL Models with Early Audio-Visual Fusion (RLDD).

Model	Class	Precision	Recall	F1-Score	Accuracy	Best Parameters
XGBoost	Deception	85.71%	92.31%	88.89%	88.00%	n_estimators = 10, max_depth = 3, learning_rate = 0.3
	Truthful	90.91%	83.33%	86.96%		
Conv1D	Deception	90.91%	92.31%	88.89%	88.00%	filters=64, dropout=0.3
	Truthful	92.31%	83.33%	86.96%		
CNN+ LSTM	Deception	80.00%	92.31%	85.71%	84.00%	filters=128, units=64
	Truthful	90.00%	75.00%	81.82%		
Decision Tree	Deception	75.00%	92.31%	82.76%	80.00%	max_depth = 5, min_samples_split = 2, criterion = 'entropy'
	Truthful	88.89%	66.67%	76.19%		
BiLSTM	Deception	78.57%	84.62%	84.62%	80.00%	units=128, dropout=0.3

Model	Class	Precision	Recall	F1-Score	Accuracy	Best Parameters
	Truthful	81.82%	75.00%	78.26%		
Random Forest	Deception	100%	53.85%	70.00%	76.00%	n_estimators = 100, max_depth = 5
	Truthful	66.67%	100%	80.00%		
BiGRU	Deception	76.92%	76.92%	76.92%	76.00%	units=128, dropout=0.3
	Truthful	75.00%	75.00%	75.00%		
Logistic Regression	Deception	75.00%	69.23%	72.00%	72.00%	C = 10, penalty = 'l2', solver = 'liblinear'
	Truthful	69.23%	75.00%	72.00%		
SVM	Deception	72.73%	61.54%	66.67%	68.00%	C = 10, gamma = 'scale'
	Truthful	64.29%	75.00%	69.23%		
ViT	Deception	62.50%	76.92%	68.97%	64.00%	head_size=32, dropout=0.2
	Truthful	66.67%	50.00%	57.14%		
GCN	Deception	46.15%	46.15%	46.15%	44.00%	Units=32, dropout=0.2
	Truthful	41.67%	41.67%	41.67%		

IV.3.4.2. Late Fusion Results

Table 18 summarizes the model performances under the late fusion setup, where predictions from separately trained audio and visual models were combined via soft voting.

Table 18 ML/DL Performance Using Late Fusion Strategy (ComParE + ViT).

Model	Class	Precision	Recall	F1-Score	Accuracy
CNN+ LSTM	Deception	100%	92.31%	96.00%	**96.00%**
	Truthful	92.31%	100%	96.00%	
Random Forest	Deception	100%	84.62%	91.67%	**92.00%**
	Truthful	85.71%	100%	92.31%	
Logistic Regression	Deception	91.67%	91.67%	91.67%	**92.00%**
	Truthful	92.86%	92.86%	92.86%	
XGBoost	Deception	100%	84.62%	91.67%	**92.00%**
	Truthful	85.71%	100%	92.31%	
BiGRU	Deception	92.31%	92.31%	92.31%	**92.00%**
	Truthful	91.67%	91.67%	91.67%	
Conv1D	Deception	92.31%	92.31%	92.31%	**92.00%**
	Truthful	91.67%	91.67%	91.67%	

Model	Class	Precision	Recall	F1-Score	Accuracy
SVM	Deception	100%	76.92%	86.96%	**88.00%**
	Truthful	80.00%	100%	88.89%	
ViT	Deception	91.67%	84.62%	88.00%	**88.00%**
	Truthful	0.8462	0.9167	0.8800	
BiLSTM	Deception	85.71%	92.31%	88.89%	**88.00%**
	Truthful	90.91%	83.33%	86.96%	
Decision Tree	Deception	90.00%	69.23%	78.26%	**80.00%**
	Truthful	73.33%	91.67%	81.48%	
GCN	Deception	41.67%	38.46%	40.00%	**40.00%**
	Truthful	38.46%	41.67%	40.00%	

Across both model types, late fusion consistently outperformed early fusion. This indicates that allowing each modality to be modeled independently before combining their predictions results in better generalization and higher accuracy. The structure of late fusion enables each model to specialize in learning the unique characteristics of its input domain (acoustic vs. visual), leading to more robust final predictions.

Figure 34 below provides a visual comparison between early and late fusion accuracy across all tested models. As shown, late fusion maintains superior performance, particularly in deep learning models such as CNN+LSTM and BiGRU.

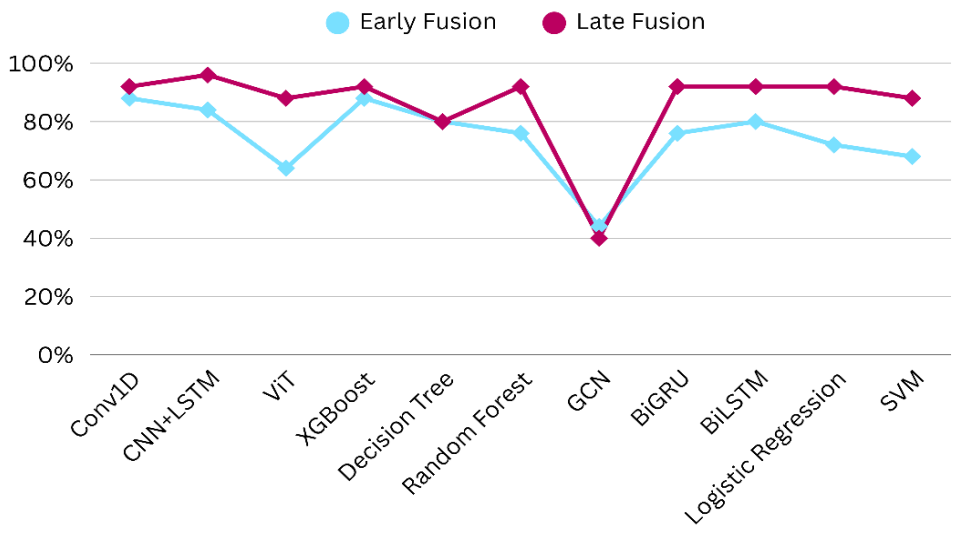


Figure 34 Early vs. Late Fusion Accuracy.

IV.4. Performance Summary and Comparison Table

This section provides a brief summary of the strongest-performing models across the different modalities tested. Instead of separating models by type, only the ones that achieved the best results for each category are shown here, whether they were based on machine learning or deep learning.

Visual models delivered the highest overall performance, especially those using ViT features with architectures like CNN+LSTM and BiGRU. In the audio setting, both SVM and Conv1D performed well with ComParE_2016 features. These results show that performance depends not just on the model itself, but also on the type of data it learns from and how well it's tuned.

The **Table 19** below highlights the top models for audio and visual inputs, along with their accuracy and key configuration details.

Table 19 Best Performing Models Across Modalities

Modality	Best Model	Accuracy	Notes
Audio	SVM	88%	C = 10, gamma = 'scale'
	Conv1D		filters=64, dropout=0.3
Visual	CNN+LSTM	96%	ViT features, DL combination model
	BiGRU		units=128, dropout=0.2

IV.5. Comparative Analysis of MFCC and ComParE_2016 Audio Features

This section presents a comparative evaluation of two audio feature types—**MFCC** (Mel-Frequency Cepstral Coefficients) and **ComParE_2016**—in the context of deception detection using CNN-based models. Each feature type was tested using a dedicated experimental setup designed to highlight its strengths and limitations in capturing deception-relevant speech cues.

The **MFCC features** were extracted using a 42-dimensional configuration from OpenSMILE and evaluated using a **Conv1D** model trained with **GELU activation**, **L2 regularization**, and **Stratified K-Fold Cross-Validation (5 folds)**. In contrast, the **ComParE_2016 feature set**, extracted using OpenSMILE's official configuration, was evaluated using the same Conv1D architecture but trained on a fixed **80/20 train-test split** with **ReLU activation**.

The goal of this comparison is to assess which acoustic representation better captures the paralinguistic and emotional markers typically associated with deceptive behavior.

Table 20 CNN Accuracy Comparison: MFCC vs. ComParE_2016.

Feature Type	Training Setup	Precision	Recall	F1-Score	Accuracy	Best Config
ComParE_2016	80/20 split, ReLU	92.31%	83.33%	87.50%	**88.00%**	filters=64, dropout=0.3
MFCC (42D)	K-Fold, GELU, L2 regularizer	76.11%	75.10%	74.95%	**75.10%**	filters=32, dropout=0.3

Based on these results, **ComParE_2016 features** clearly outperformed **MFCC** in all key evaluation metrics. While MFCCs effectively capture linguistic and phonetic content, they tend to overlook the **emotional, stress-related, and paralinguistic aspects** of speech that are crucial in deception detection. On the other hand, ComParE_2016 is explicitly designed to capture such cues, including vocal tension, pitch variation, and speech dynamics, making it better suited for detecting deceptive behavior in real-world settings.

IV.6. Result Analysis and Discussion

This section provides a critical analysis of the performance observed across all experimental settings. The aim is to consolidate findings and highlight how model architectures, feature types, and fusion strategies impacted deception detection accuracy.

IV.6.1. Performance Comparison Across Models and Techniques

Model performance varied across modalities, but a few patterns stood out. In the audio experiments, both SVM (in the machine learning group) and Conv1D (in the deep learning group) achieved the highest accuracy at 88%, showing that well-tuned classical models can still compete with deep learning when the input features are well selected — in this case, **ComParE_2016**.

On the visual side, deep learning models clearly outperformed classical ones. While several machine learning models like SVM, Logistic Regression, and XGBoost each reached 88% accuracy, the CNN+LSTM and BiGRU models achieved significantly higher results at 96%, especially when trained on ViT-based frame embeddings. These architectures seemed particularly effective at capturing the complex spatial patterns and subtle facial cues relevant to deception detection.

When it came to combining modalities, models using **late fusion** consistently produced the strongest outcomes. The CNN+LSTM model, when applied in a late fusion setup, achieved 96% accuracy — matching its visual-only performance but with stronger class-level consistency. This reinforced the idea that learning from each modality independently and merging predictions afterward offers a more balanced and robust approach than early feature-level fusion.

IV.6.2. Impact of Dataset Type and Feature Extraction

The RLD dataset enabled robust evaluation thanks to its real-life trial setting and synchronized audio-visual content. Among audio features, ComParE_2016 outperformed MFCC, as it better captured prosodic and emotional elements relevant to deceptive speech. Visual features extracted using ViT embeddings proved highly discriminative, supporting the hypothesis that behavioral micro-expressions are essential deception cues. Text-based features, while explored using BERT, offered limited benefit, reinforcing the observation that *how* something is said matters more than *what* is said in deceptive contexts.

IV.6.3. Discussion of Overfitting and Generalization

Model generalization was addressed through both fixed 80/20 train-test splits and K-Fold cross-validation, depending on the experimental setup. While most models showed

stable learning behavior, overfitting was observed in certain deep learning configurations, often indicated by divergence between training and validation curves. These issues were mitigated through the use of dropout, careful tuning of network depth and units, and in some cases switching from ReLU to GELU activation functions. Overall, late fusion strategies led to better generalization, as they allowed each model to specialize in its respective modality before combining predictions.

IV.6.4. Lessons Learned and Research Insights

Several key insights emerged throughout the study. First, deep learning models tend to require large, high-quality feature sets to perform optimally. When tested with MFCC features, model accuracy was lower, likely due to the limited emotional and prosodic content captured. In contrast, **ComParE_2016 features** proved more effective at capturing deception-relevant vocal cues.

Another important lesson was the value of proper tuning. When we used Grid Search to adjust parameters — like the number of units or dropout rates — models that didn't perform well at first showed major improvements. This showed us how much of a difference good parameter selection can make, especially in deep learning.

We also learned that fusion methods matter. Late fusion — where audio and visual models are trained separately and their predictions are combined — worked better than early fusion in most cases. This makes sense, since each model gets to focus on its own type of data before combining.

Text-based detection turned out to be less helpful than expected. Even though we used BERT, the results were lower than audio and visual models. This confirmed our hypothesis that how someone speaks — tone, stress, and emotion — is more telling than what they say.

Overall, this project taught us that the combination of good features, smart tuning, and the right fusion approach can make a big difference in detecting deception. It also showed us the limits of some tools like MFCC or text-based analysis, helping us better understand where to focus on future work.

IV.6.5. Comparison with Prior Work

To evaluate the effectiveness and originality of our approach, we compare our system's results with prior studies that used the same or similar datasets and modalities. Three key works were selected for this comparison: the foundational Real-Life Trial dataset study by Pérez-Rosas et al. [83], the preprint by Farzeenak et al. (2024) that directly uses the RLDD dataset , and our own results.

We compare based on the following aspects:

- **Modality**: Which data types are used (Audio, Visual, Text).
- **Feature Extraction Techniques** : e.g., OpenSMILE (ComParE), ViT, BERT, etc.
- **Model Type**: Classical ML (e.g., SVM), DL (e.g., CNN+LSTM, BiGRU), or Transformers.
- **Fusion Strategy**: Early fusion, late fusion, or unimodal pipelines.
- **Reported Accuracy**: Final reported classification accuracy, if available.

Table 21 Comparison with Prior Work.

Study	Modality	Feature Extraction	Models Used	Fusion Strategy	Accuracy
Pérez-Rosas et al. (2015) [64]	Audio, Visual, Text	Handcrafted features	SVM, RF	Early fusion	~75%
Farzeenak et al. (2024)	Visual (frame-based)	Manual Gesture Annotation	CNN, LSTM	Late fusion	~90% (visual only)
This Study (2025)	Audio, Visual, Text	ComParE (OpenSMILE), ViT, Whisper + BERT	SVM, BiGRU	**Late fusion**	**96%**

A Note on Manual Gesture Annotation:

Manual gesture annotation refers to the process of human annotators labeling specific gestures (e.g., hand movements, facial cues) in video frames before feeding them to a model. This is typically more labor-intensive and subjective, but it provides models with clean labeled visual features such as smiles, head tilts, or fidgeting.

By contrast, our system uses **Vision Transformers (ViT)** to automatically extract features from raw frames without manual intervention. While ViT may not explicitly label expressions like a human would, it learns spatial patterns—including facial micro-expressions and movement patterns—from the pixel distributions over time. Thus, our method is more scalable and suitable for real-time or practical deployment, without requiring gesture-by-gesture annotation.

IV.7. Final Model Decision

IV.7.1. Evaluation Criteria for Model Selection

When candidate models achieve equal or comparable accuracy, accuracy alone is no longer a sufficient basis for deployment. To ensure the most reliable, efficient, and explainable model is selected, we adopted a multi-criteria evaluation framework. This framework is applied consistently across both the audio and visual modalities.

The criteria used in our comparative evaluation are:

a) **Model Complexity and Inference Time** Measures how computationally expensive the model is and how fast it can produce predictions. Simpler models tend to be more efficient and easier to deploy on limited hardware.

b) **Explainability and Interpretability** Reflects how transparent the model's internal decision-making process is. Interpretable models are easier to validate and trust in sensitive applications.

   To support model interpretability, we used **LIME (Local Interpretable Model-Agnostic Explanations)** [85] to visualize which features contributed most to each prediction. LIME was selected for its ability to provide instance-level explanations, especially useful when analyzing high-dimensional feature sets like **ComParE 2016 audio features and ViT visual embeddings**.

In a sensitive application like deception detection, transparency is critical. By showing why a prediction was made—whether it labels someone as truthful or deceptive—LIME helps users and stakeholders understand the system's logic, which is essential for trust, fairness, and responsible use.

c) **Feature Dependency** Assesses how dependent the model is on complex or high-dimensional input features. Simpler models tend to be more adaptable and robust across different feature types.

d) **Overfitting Behavior and Stability** Evaluates the model's generalization ability. A model that performs well on training data but poorly on validation data is likely overfitting and unsuitable for deployment.

e) **Training Stability** Refers to the model's consistency across multiple training runs. Models that are stable are more predictable and reproducible.

f) **Resource Efficiency** Takes into account the hardware and runtime requirements of the model during both training and deployment phases.

These criteria guided our analysis in selecting the final audio and visual models described in the following sections.

IV.7.2. Comparative Evaluation of Audio Models

As presented earlier in Table 19, the two best-performing models in the audio modality were SVM and Conv1D, each achieving 88% accuracy. However, to determine which model is most suitable for deployment, we conducted a deeper comparison based on the following criteria: model complexity, training stability, inference time, explainability, overfitting behavior, and feature dependency.

a. **Model Complexity and Inference Time:** Conv1D, as a deep learning model, involves multiple layers, learnable filters, and higher computational cost. In contrast, SVM is a lightweight classical model with fewer parameters and significantly faster inference — especially valuable for real-time or embedded deployment.

b. **Explainability and Interpretability:** SVM offers a high degree of interpretability due to its transparent decision boundaries and relatively simple mathematical formulation. As a classical machine learning model, it is inherently more explainable

than deep learning architectures like Conv1D, which involve multiple hidden layers and complex learned representations. This makes SVM a preferable choice in scenarios where model transparency and explainability are important.

c. **Feature Dependency:** Although SHAP is often used to interpret complex models, we could not apply it here due to the use of the ComParE 2016 feature set, which includes over 6000 features. Visualizing this many variables in a SHAP summary would be ineffective. Nevertheless, since SVM is already interpretable and LIME results were sufficient, SHAP was not required.

d. **Overfitting Behavior and Stability:** SVM demonstrated stable learning behavior, maintaining a consistent gap between training and validation accuracy with no signs of overfitting. Despite enhancing the feature space by using ComParE 2016—far more comprehensive than eGeMAPS—Conv1D still showed signs of variance during training. This outcome highlights the strength of classical models like SVM, which can generalize effectively with limited training data, unlike deep learning models that typically require larger datasets.

IV.7.3. Selected Audio Model and Rationale

Following the comparative evaluation, **SVM** was selected as the audio model to be deployed in the final deception detection system. While Conv1D achieved the same accuracy, SVM offered a more stable and interpretable solution that generalizes well with limited data. Its performance with the high-dimensional ComParE 2016 feature set was both efficient and reliable.

The training process confirmed this reliability. The selected configuration (C = 10, gamma = 'scale') showed no signs of overfitting and achieved stable convergence, as shown in the overfitting plot below.

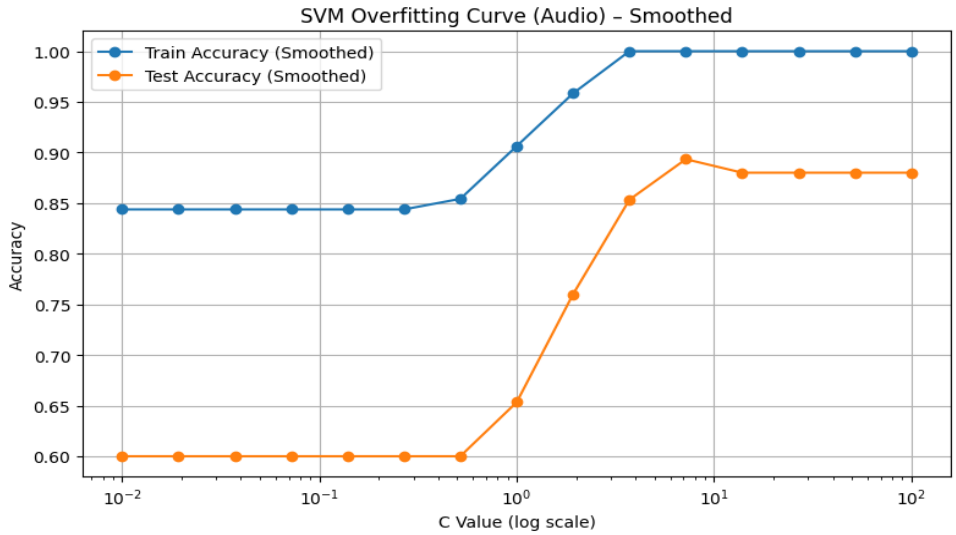


Figure 35 Overfitting curve for SVM (C = 10).

In addition to its robust performance, SVM also allows for interpretability. To better understand the decision process, we applied **LIME** (Local Interpretable Model-Agnostic Explanations). The output highlighted the top contributing features in both truthful and deceptive predictions.

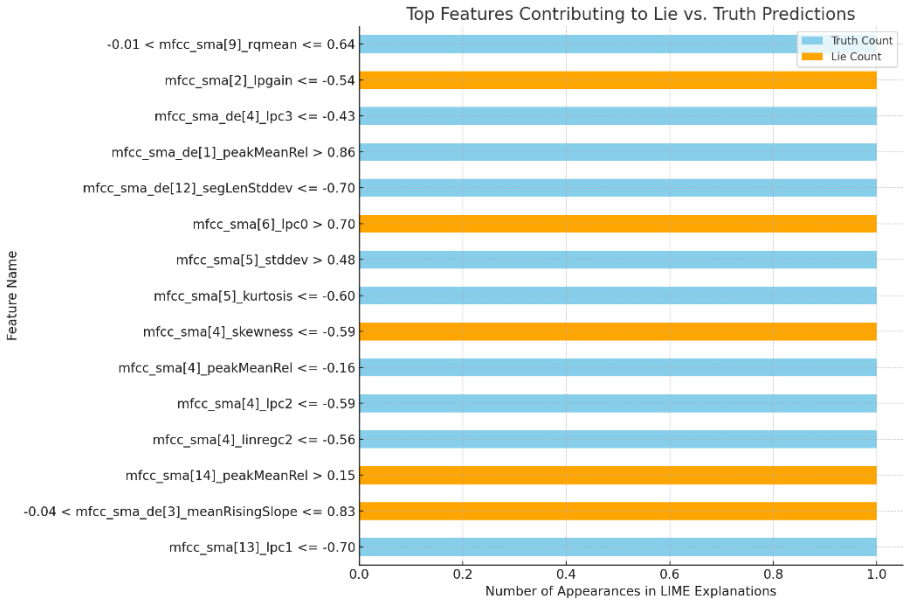


Figure 36 Top LIME features contributing to SVM predictions (combined).

The following examples illustrate the LIME outputs for two specific cases—one where the model predicted deception, and another where it predicted truth. These results confirm that SVM's decisions were based on consistent and relevant feature patterns.

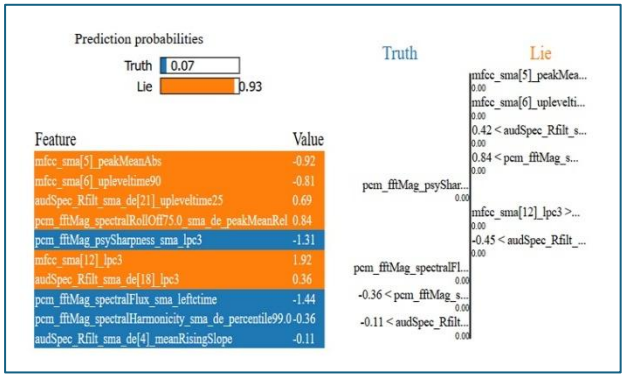


Figure 37 LIME explanation for a deception prediction SVM.

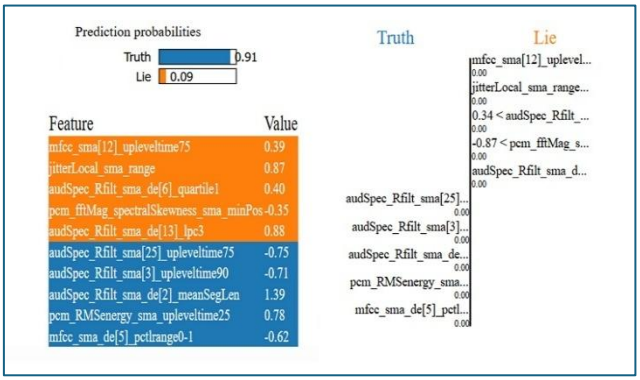


Figure 38 LIME explanation for a truth prediction SVM.

In conclusion, SVM was chosen not only for its competitive accuracy but also for its simplicity, interpretability, and stable performance. These qualities make it an ideal candidate for integration into the audio branch of the deception detection system.

IV.7.4. Comparative Evaluation of Visual Models

In the visual modality, both **BiGRU** and **CNN+LSTM** achieved the highest classification accuracy of **96%**. However, due to the identical performance, we applied the same evaluation criteria—model complexity, explainability, overfitting behavior, training stability, feature dependency, and resource efficiency—to determine which model was more appropriate for deployment.

a. **Model Complexity and Inference Time:** CNN+LSTM is a combination of two architectures: a convolutional front-end and a recurrent back-end. This inherently increases model complexity, memory usage, and inference latency. In contrast, BiGRU is a single, unified recurrent model, significantly simpler in structure and more efficient in execution.

b. **Explainability and Interpretability:** While neither model is inherently interpretable, BiGRU's simplicity offers a marginal advantage in transparency. Furthermore, we attempted to apply SHAP on the BiGRU model using ViT-extracted features.

However, the visual output was sparse and inconclusive, offering limited interpretive insight. For this reason, we did not rely on SHAP for the final explanation.

c. **Feature Dependency:** Both models were trained on visual features extracted using the Vision Transformer (ViT). These high-dimensional embeddings were suitable for deep architecture. However, BiGRU was able to interact with them in a more stable and predictable manner, requiring fewer processing layers to extract temporal dynamics.

d. **Overfitting Behavior and Stability:** BiGRU displayed a smooth and consistent convergence trend throughout training. The validation accuracy increased steadily alongside the training accuracy, showing no clear signs of overfitting. By contrast, CNN+LSTM exhibited significant fluctuations in validation accuracy despite achieving high training accuracy—suggesting a potential overfit to the training data.

e. **Training Stability:** BiGRU showed consistent results across runs, with low variance in performance. CNN+LSTM, however, was more sensitive to initialization and training conditions, occasionally resulting in inconsistent outcomes.

f. **Resource Efficiency:** Given that BiGRU is a lighter model with fewer parameters, it was more efficient in terms of memory and training time. Its lower computational demand makes it better suited for scalable or real-time applications.

IV.7.5. Selected Visual Model and Rationale

Considering the evaluation across all six criteria, **BiGRU** was selected as the final visual model for the deception detection system. While CNN+LSTM achieved the same accuracy, it suffered from unstable validation behavior and significantly higher complexity.

The smoothed accuracy plot for BiGRU **Figure 39** shows stable and parallel improvement in training and validation accuracy over epochs, further supporting its generalization strength. This contrasts with the CNN+LSTM training curve **Figure 40**, which displays notable oscillations in validation accuracy—suggesting overfitting and less reliability

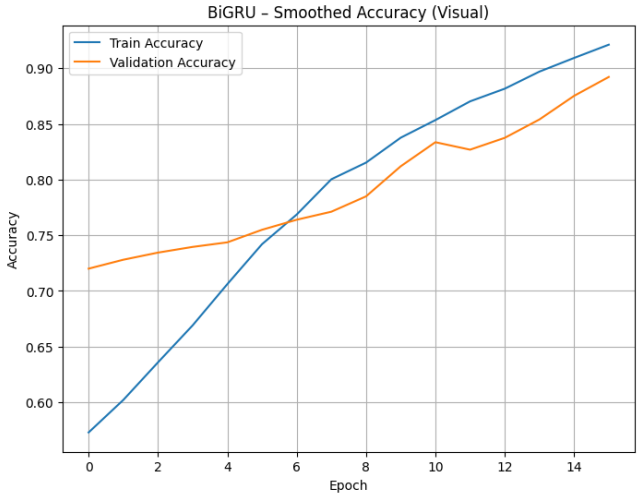


Figure 39 BiGRU – Smoothed Accuracy Plot (Visual).

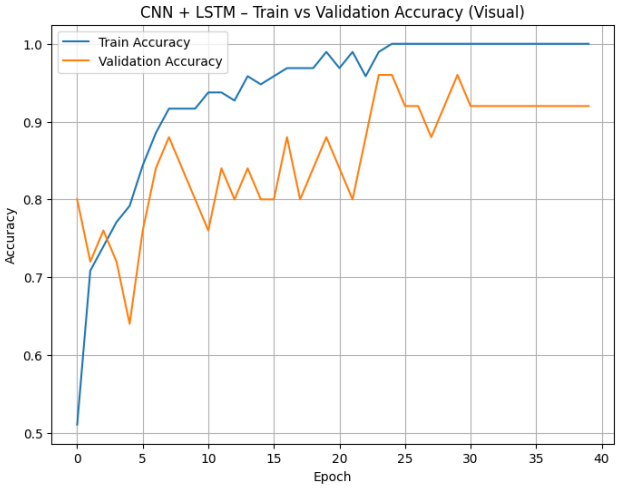


Figure 40 CNN+LSTM – Training vs Validation Accuracy (Visual).

Although deep learning models are generally less interpretable than classical models, we applied **LIME** (Local Interpretable Model-Agnostic Explanations) to visualize which features most influenced BiGRU's predictions. The results confirmed that the model's decisions were based on consistent and meaningful patterns derived from the ViT visual embeddings.

These visualizations reinforced the interpretability of the selected model and provided additional confidence in its deployment.

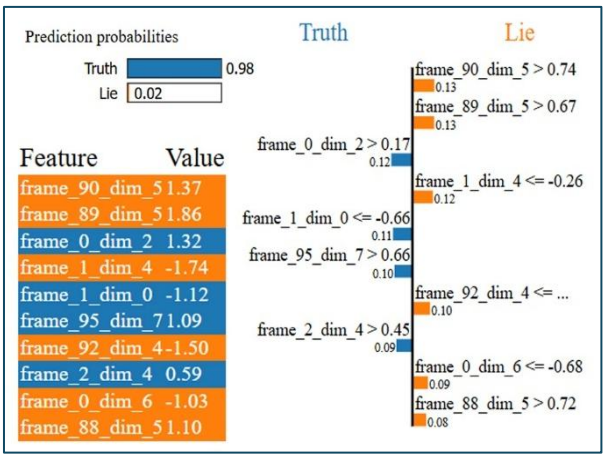


Figure 42 LIME explanation for a truth prediction BiGRU.

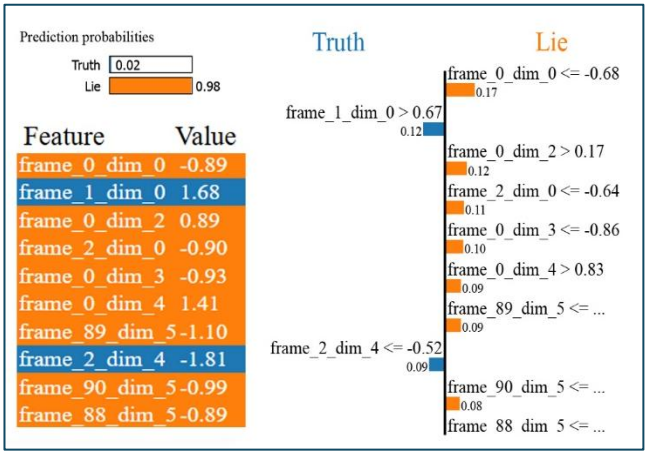


Figure 41 LIME explanation for a deception prediction BiGRU

In summary, BiGRU was selected for its **simplicity, stability, and lower computational complexity**, all while maintaining the highest possible performance. These qualities made it the most appropriate choice for deployment in the visual branch of the system.

IV.8. Conclusion

The results presented in this chapter demonstrate the effectiveness of modality-specific modeling and late fusion in deception detection. Classical models such as SVM achieved competitive results with high-dimensional features, while deep learning models like BiGRU and CNN+LSTM showed superior performance in visual and fused settings. Late fusion consistently outperformed early fusion, confirming its value in multimodal learning. The final models were selected based on accuracy, stability, and interpretability, ensuring both reliable performance and suitability for real-world use.

Chapter V. Final Implementation and Deployment

V.1. Introduction

This chapter focuses on the practical implementation of the final deception detection system. After selecting the models most suited for deployment, the chapter introduces the technical tools used to build and run the system. It then outlines how the system components work together, from recording to prediction and result display. In addition to technical design, attention was also given to physical presentation and certain psychological aspects that influence how the subject interacts with the setup. The chapter ends with a full walkthrough of how the system is used in a real session.

V.2. Tools, Languages, and Libraries Used

V.2.1. Programming Language and Environment

The final deception detection system was implemented and executed locally using **Python 3.10** on a personal laptop (HP EliteBook x360 1030 G8) equipped with **16 GB RAM**, an **Intel Core i5-1135G7 processor (11th Gen, 2.40 GHz)**, and **Intel Iris Xe integrated graphics**. Development and deployment were carried out using **PyCharm**, while the trained models (originally developed and exported from Google Colaboratory) were integrated into a custom-built interface.

The backend system includes modules for recording webcam and microphone input, extracting audio and visual features, and running real-time predictions using pre-trained models (SVM for audio and BiGRU for visual). The frontend interface was developed using **HTML, JavaScript, and Flask**, and designed for interviewer-only access, keeping the subject unaware of the UI to preserve natural behavior.

Although the final backend reuses core libraries used in model training (e.g., TensorFlow, Torch, OpenCV, OpenSMILE), only the essential components required for **inference and UI integration** were included to minimize overhead and enhance responsiveness.

V.2.2. Core Libraries and Frameworks

The final application relied on a targeted set of Python libraries to support audio-visual processing, real-time inference, and user interaction:

- **Flask** – Provided the backend web server used to handle video uploads and model predictions.
- **OpenCV** – Used for real-time face detection and frame extraction from recorded videos.
- **OpenSMILE** – Applied to extract ComParE_2016 audio features from recorded microphone input.
- **Joblib** – Enabled loading of the serialized SVM model trained on audio features.
- **PyTorch** – Used to load and run the pre-trained BiGRU model for visual predictions.
- **Transformers (Hugging Face)** – Provided the Vision Transformer (ViT) feature extractor used for frame embeddings.
- **NumPy** – Supported numerical operations and feature handling.
- **MoviePy** and **FFmpeg** – Assisted in extracting and converting audio and video segments from user recordings.
- **Playsound** – Used to trigger feedback audio (e.g., "Liar" or "Truth") based on model output.

This focused library set ensured efficient processing while preserving compatibility with previously trained models.

V.3. Detection System Architecture

The deception detection system is composed of two primary layers: a backend that handles video/audio processing and prediction, and a frontend that serves as a hidden control panel for the interviewer. These components communicate through a Flask server and a browser-based interface, working together to produce real-time deception predictions.

V.3.1. System Overview

The system is built around a client-server model. The **frontend** is a web interface (index.html) loaded locally in a browser, while the **backend** is powered by a Python Flask

application (`app.py`). When the interviewer initiates a recording session through the UI, the browser captures webcam and microphone input. This video is then sent via an HTTP POST request to the `/predict` route of the Flask backend.

The backend receives the recording, extracts audio and visual features, runs predictions using pretrained machine learning and deep learning models (SVM and BiGRU), and sends the result ("LIAR" or "TRUTH") back to the frontend. The result is displayed on-screen along with sound effects to enhance the experience.

Importantly, the frontend is not visible to the subject. This hidden setup ensures the authenticity of reactions and prevents behavioral masking. The diagram below summarizes the full data flow from user input to model prediction and system response.

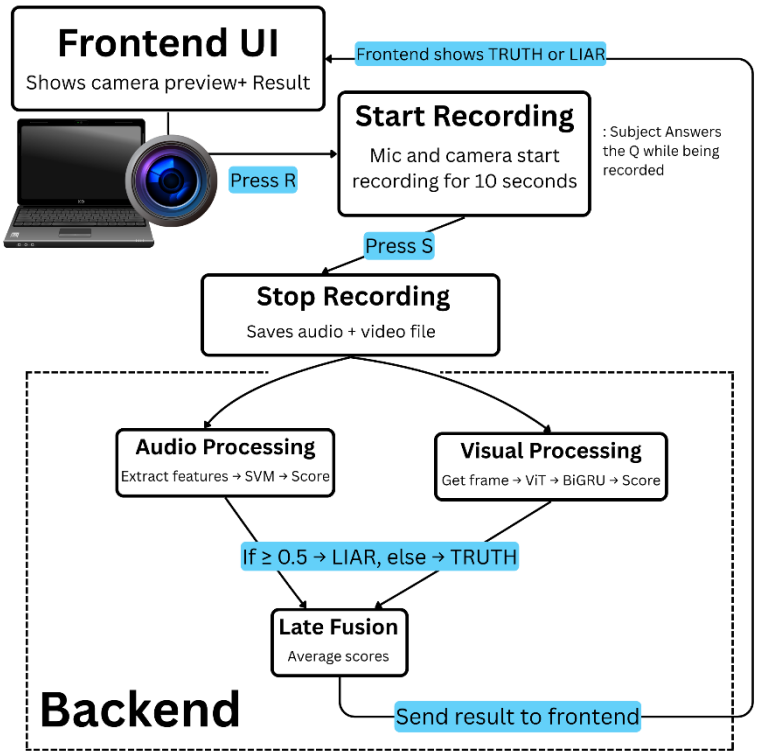


Figure 43 Workflow diagram during deception detection with LieBusters..

V.3.2. Backend: Feature Extraction and Prediction Pipeline

The backend (app.py) is responsible for recording input, processing it, and generating a final deception prediction. After training the models on Google Colab, the final versions

were exported and downloaded as .pkl and .h5 files, then integrated into the local backend system for prediction. Its components include:

- **Recording**: Using ffmpeg, the backend captures a 10-second video clip from an external webcam and microphone. The video is saved locally for analysis.
- **Audio Extraction**: The audio stream is extracted from the video using ffmpeg and processed with **OpenSMILE** to obtain **ComParE 2016 functionals**. These features (6373-dim) are standardized using a previously saved scaler and then passed to a pretrained **SVM model**.
- **Visual Extraction**: The middle frame of the recorded video is extracted using OpenCV. A pretrained **Vision Transformer (ViT)** from Hugging Face generates embeddings, which are scaled and reshaped to match the input required by a trained **BiGRU** model.
- **Prediction**:
  - The audio model outputs the probability of deception from the SVM.
  - The visual model (BiGRU) outputs another probability.
  - These are averaged, and if the result exceeds 0.5, the prediction is "LIAR"; otherwise, "TRUTH."
- **Response**: The prediction (along with raw scores) is returned to the frontend in JSON format.

This backend pipeline maintains the exact same preprocessing logic used during model training, ensuring consistency in the system's behavior and results.

V.3.3. Frontend: Hidden Interface and Control Panel

The frontend is defined in index.html and includes HTML, CSS (inline), and JavaScript. It features:

- A **dark, glitchy horror-style interface**, with a bold red-on-black color scheme and a flickering “ARE YOU LYING?” title rendered using a spooky font (Creepster)
- A **live mirrored webcam preview**

- Keyboard-based controls:
  - Press **R** to start recording
  - Press **S** to stop and trigger analysis
- A **status area** that displays live messages and the final verdict ("🟥 LIAR!" or "🟩 TRUTH!")
- Integrated **sound effects**:
  - Heartbeat background
  - Distinct audio for “LIAR” and “TRUTH” predictions

Importantly, this interface is not shown to the person being interviewed. The interviewee only sees the camera inside the physical box (see Section IV.5), not the interface or any feedback. This was a deliberate psychological decision to avoid response bias. Subjects knowing they are being evaluated—especially while seeing their face or feedback—may unconsciously or intentionally alter their behavior. By restricting the interface to the interviewer, the system encourages natural emotional and behavioral cues, improving detection reliability.

V.4. Visual Identity and Psychological Framing

Beyond technical accuracy, the deception detection system was deliberately designed to evoke emotion, tension, and subtle psychological influence. From its name to its visual style, every element of the interface and interaction was crafted to heighten the emotional charge of the experience—while preserving the integrity of behavioral cues.

V.4.1. Brand Identity: Naming and Conceptual Framing

The system was named **LieBusters** as a play on the familiar and culturally loaded title “Ghostbusters.” The name carries immediate emotional weight—suggesting confrontation, exposure, and consequence. It implies that deception, like a ghost, can be detected and "busted" by the system.

Psychologically, the name is designed to create a **sense of pressure** and unease in the subject. While the subject does not see the interface or the branding directly, the branding

still frames the system's tone and purpose, setting the expectation for a serious evaluation process. The title “LieBusters” supports the idea that this is not a neutral interaction—it is an interrogation, one where the truth will be extracted.

V.4.2. Visual Identity: Logo, Fonts, and Color Palette

The visual identity of *LieBusters* was inspired by a blend of pop culture and psychological symbolism. The logo draws from the iconic *Ghostbusters* emblem, combined with the figure of *Pinocchio*—a well-known symbol of lying—capturing the system’s core mission of exposing deception. The design concept positions the system as one that “catches Pinocchios.” The chosen color palette emphasizes dominance and intensity, with red and black conveying assertiveness, boldness, and a sense of control. These visual elements were selected to evoke feelings of tension and seriousness in the environment. The user interface complements this identity with a dark, mystique-driven aesthetic that suggests surveillance and secrecy. Altogether, the identity aims to reinforce the emotional weight of the experience and subtly influence behavior through visual and symbolic cues.

Figure 44 Visual Identity of Lie Busters.

V.4.3. Environmental Control and Psychological Strategy

The environment of the *LieBusters* system was deliberately designed to influence the subject's emotional state through controlled visual and auditory elements. The user interface is hidden from the subject to prevent them from consciously altering their behavior. In addition, background audio such as a slow, continuous heartbeat was included to induce subtle anxiety and increase psychological pressure during recording. Final prediction sounds were carefully chosen to reinforce emotional responses: the "truth" sound is soft and affirming, while the "lie" sound resembles a buzzer, evoking discomfort or a sense of error. These elements work together to create a controlled setting that encourages natural reactions and discourages performance or deception.

V.5. Hardware Deployment and System Assembly

V.5.1. Hardware Components and Custom Mount

The hardware configuration of the *LieBusters* system was designed for portability, psychological subtlety, and visual integration. Two primary components were involved in the assembly: a compact surveillance camera and a custom-manufactured laptop-mounted extension.

V.5.1.1. 1. Surveillance Camera

The chosen recording device was the LMNOOP Original SQ11 1080P Mini Infrared Surveillance Camera, selected for its compact form factor, night vision capability, and unobtrusive design. This camera operates as a portable digital video recorder and was ideal for maintaining a concealed, controlled observation setup. Key specifications include:

- Resolution: 1080p
- Infrared night vision
- Rechargeable battery
- Estimated price: $59.18 USD

Figure 45 SQ11 1080P Mini

V.5.1.2. Custom PMMA-Mounted Extension

To house and control the camera, a laptop-mounted frame was built using **PMMA (polymethyl methacrylate)** in a **shell white color**. The extension was **laser-cut and assembled in a private workshop** equipped with fabrication tools. It is attached to the top edge of the laptop screen and features a **rotating pin mechanism**, allowing manual adjustment of the camera's angle without requiring the subject to move. This design ensures that the interviewer can easily center the camera on the subject, improving visual consistency across recordings.

The mount is visually integrated with the system's branding through **adhesive overlays** (autocollants) that include the project's logo and red accents, reinforcing the *LieBusters* identity while visually concealing the functional elements of the camera.

Figure 46 3D Model of Custom Laptop-Mounted PMMA Camera Extension.

This setup achieves both functional accuracy and aesthetic cohesion, ensuring that the deception detection system remains discreet, controllable, and visually aligned with its psychological purpose.

V.5.2. Integration with Detection System

The camera and custom mount were directly integrated with the *LieBusters* system. The camera connects to the laptop running the backend, and recordings are triggered either from the keyboard or the frontend interface. The rotating pin allows the interviewer to adjust the camera angle manually, so the subject can stay seated naturally without being asked to move or reposition. This improves subject comfort and ensures proper framing for visual analysis without disrupting the test environment. The setup was tested to align with the visual input requirements of the ViT model, ensuring consistent frame capture.

V.5.3. Photo and Diagram of the Final Setup

The final hardware setup integrates the laptop, custom PMMA-mounted camera extension, and supporting interface in a compact and controlled environment. The images below illustrate the full configuration as it appears during deployment. The mount is fixed to the top of the laptop screen, with the camera positioned centrally for optimal subject framing. The rotation mechanism allows subtle adjustment by the interviewer without requiring the subject to move.

The physical presentation aligns with the visual identity of the system, with red accent overlays and minimal design distractions. This ensures that the detection process remains discreet, technically precise, and psychologically neutral for the subject.

Figure 47 Final Setup.

V.6. Full System Demonstration

This section outlines how the complete *LieBusters* system operates in practice, from start to finish. It includes the sequence of steps carried out during a typical deception detection session and visual examples of the system in action.

V.6.1. Step-by-Step Usage Flow

1. Setup and Positioning

The laptop is placed on a table, with the camera mount positioned and rotated to face the subject naturally. The subject is unaware of the system interface and only sees the camera.

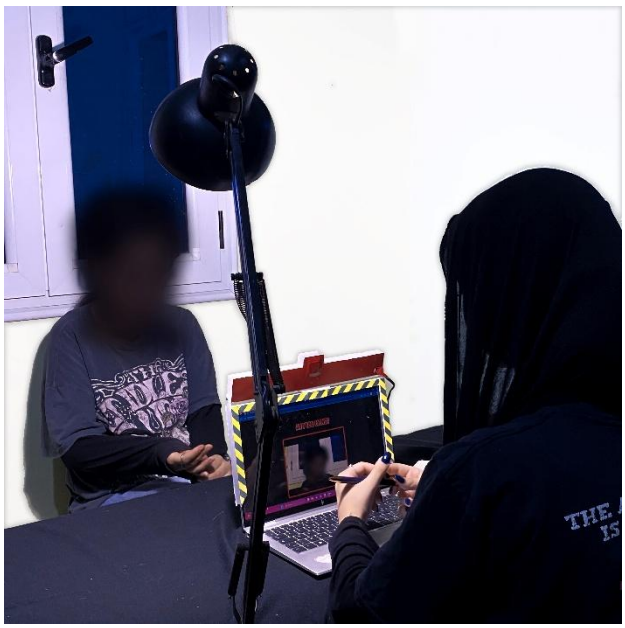


Figure 48 System Setup and Subject Positioning.

2. Recording Initiation

The interviewer launches the browser-based frontend and presses R to begin recording. A heartbeat sound plays in the background, enhancing psychological tension.

3. Subject Response

While recording, the subject is asked a specific question intended to elicit a potentially deceptive or truthful response.

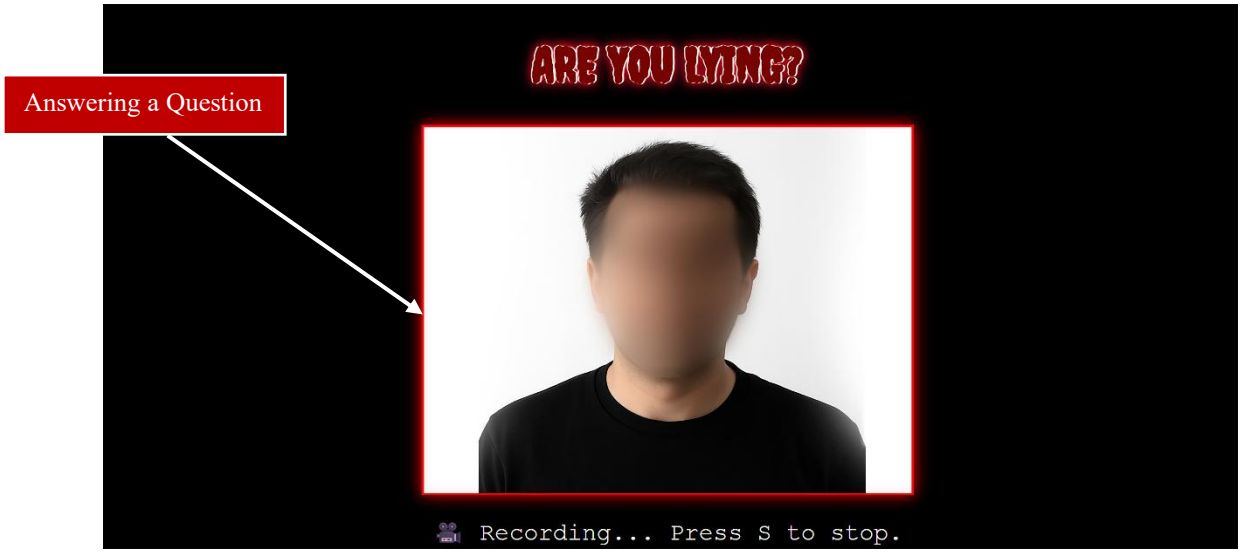


Figure 49 Subject Responding During Live Session.

4. Stopping the Recording

The interviewer presses S to stop the session. The video is automatically sent to the backend for processing.

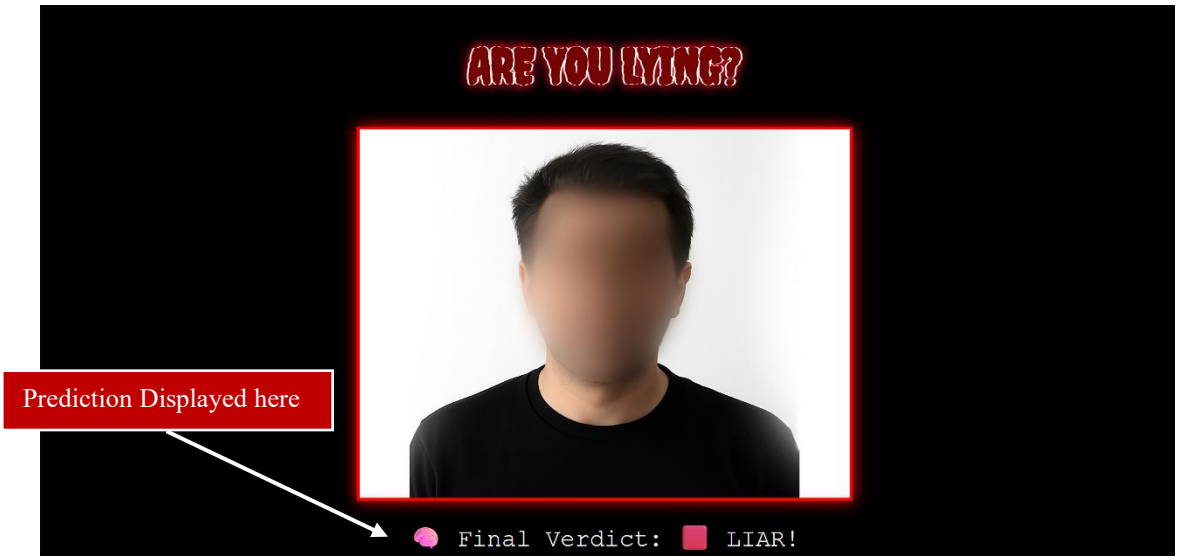
Figure 50 Recording Stopped and Sent to Backend.

5. Feature Extraction and Prediction

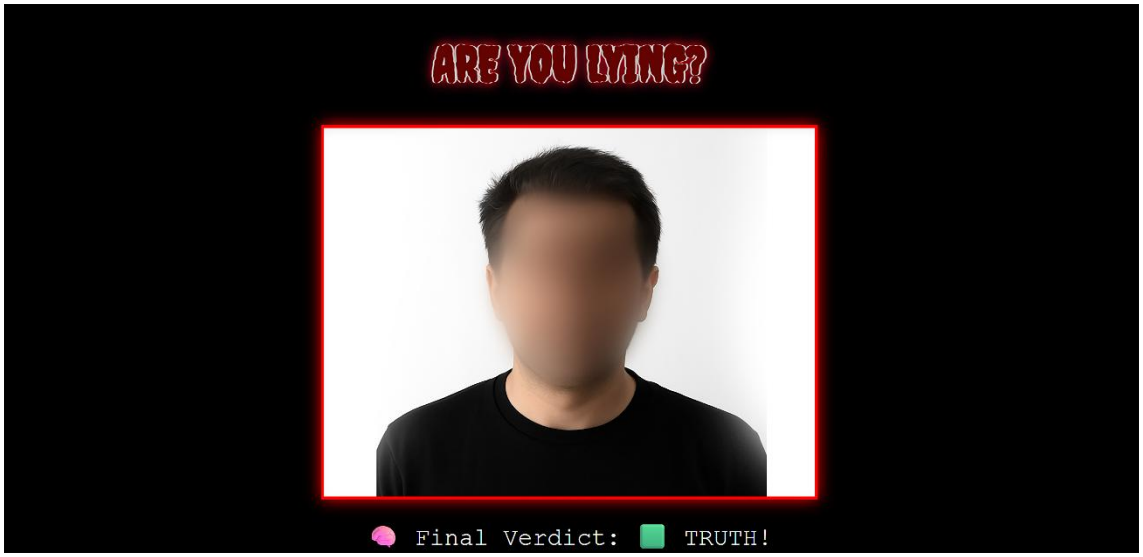
The system extracts audio and visual features, scales them, and uses pretrained models (SVM for audio, BiGRU for visual) to generate a final prediction.

6. Result Output

The frontend displays either "🟥 LIAR!" or "🟩 TRUTH!" along with a corresponding sound. This feedback is visible and audible only to the interviewer.



(a) Lie prediction output.



(b) Truth prediction output.

Figure 51 Final verdicts displayed on the frontend interface (a) Truth (b) Lie.

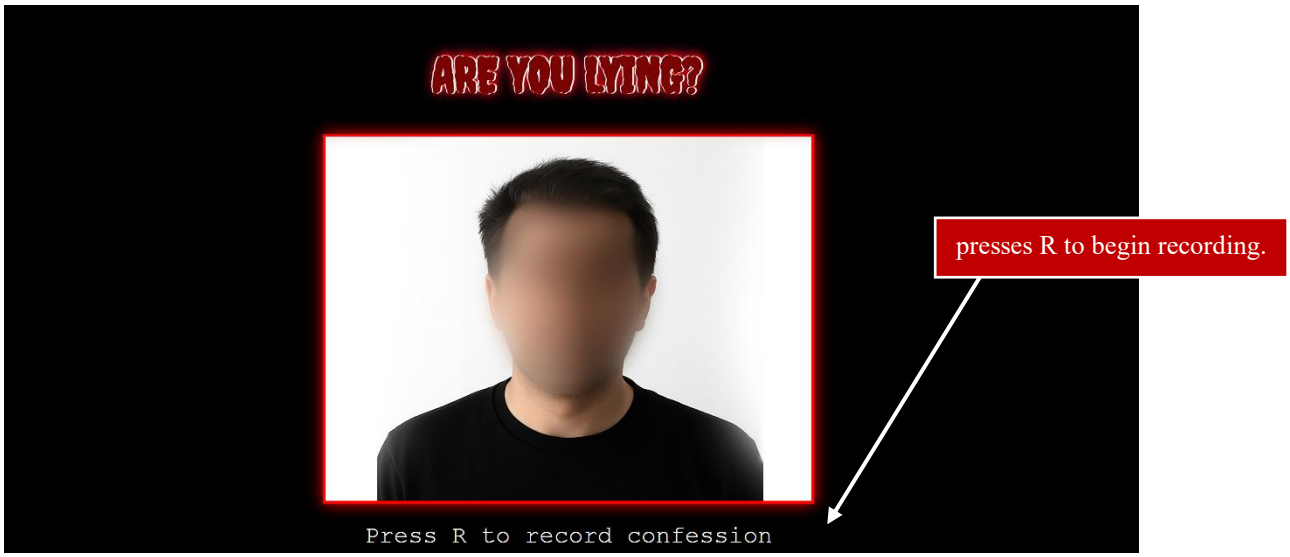


Figure 52 Recording Initiated from Frontend Interface.

V.7. Reflections and Future Work

V.7.1. Challenges and Unstable Behavior

The development of the *LieBusters* system presented several technical and environmental challenges, many of which stemmed from limited resources and real-world testing conditions:

- **Limited Dataset Availability**: The small number of publicly available deception datasets made training and generalization more difficult.
- **Hardware Limitations**: The system was tested using a standard laptop, USB webcam, and built-in microphone, which affected input quality.
- **Low-Quality Visual Input**: Inconsistent lighting and background interference occasionally reduced facial clarity.
- **Uncontrolled Recording Environments**: Noise, shadows, and background movement introduced variability into audio and video recordings.
- **Microphone and Audio Issues**: Background noise and echo, combined with the use of an internal mic, lowered audio clarity in some sessions.
- **Camera Positioning**: Without a fixed setup, framing varied across recordings. Some subjects leaned out of frame or moved unpredictably.

- **Recording Synchronization**: Ensuring consistent saving of synchronized audio and video required careful handling of formats and timing.
- **Lack of Controlled Setup**: Ideally, future versions should include neutral backgrounds, ring lights, and high-fidelity microphones to enhance consistency.

V.7.2. Potential Extensions

The current system provides a working proof of concept. However, several opportunities exist for improvement and expansion:

- **Real-Time Processing**: Implementing live analysis would remove delays and support instant detection.
- **Web Deployment**: Hosting the backend on platforms like AWS or Heroku could allow remote use via browsers.
- **Mobile App Development**: A lightweight mobile version could enable on-the-go detection using phone hardware.
- **Larger and Diverse Datasets**: Including more languages, cultures, and contexts would improve model robustness and fairness.
- **Improved Equipment**: Using better cameras, microphones, and lighting would enhance feature quality and model accuracy.
- **New Modalities**: Future systems could integrate physiological signals (e.g., heart rate, eye movement) for deeper insight.
- **Enhanced UI and Feedback**: Interface improvements such as progress bars or summary reports could benefit users, especially in formal settings.
- **Ethics and Data Privacy**: Any future deployment must include ethical safeguards and compliance with privacy regulations such as GDPR.

V.7.3. Closing Remarks

This project was both technically demanding and creatively engaging. It involved not just model development, but also real-world considerations—such as how users behave when observed, and how environments affect data collection.

The resulting prototype demonstrates that a low-cost, multimodal deception detection system is achievable. With better datasets, improved hardware, and further iteration, *LieBusters* could evolve into a practical tool for behavioral analysis in fields such as security, psychology, or law enforcement.

V.8. Conclusion

This chapter showed how the LieBusters system was fully put together—from choosing the models, to building the interface, untill testing it in real situations. The system brings together machine learning, deep learning, audio-visual processing, and psychological design choices to create a complete, working prototype. Every detail—from the model predictions to the environment setup—was considered to make sure the system felt realistic and could be used naturally. The final version runs smoothly and gives clear results, which makes it a strong base for future improvements or real-world use.

General Conclusion

The detection of deception remains one of the most intricate and socially impactful challenges in behavioral analysis. While traditional tools such as polygraph testing, voice stress analysis, and non-verbal behavior interpretation are still used, their lack of objectivity, scalability, and reliability has drawn growing criticism. This thesis addressed these shortcomings by leveraging artificial intelligence—particularly machine learning and deep learning—to automatically detect deception using multimodal data under real-world conditions.

The research was conducted on the Real-Life Trial Deception Dataset (RLDD), which contains authentic courtroom testimonies labeled as truthful or deceptive. Audio data was processed using MFCCs and the ComParE_2016 feature set, while visual data was embedded using frame sampling with a pre-trained Vision Transformer (ViT). Several machine learning and deep learning models were trained and evaluated across different input settings: audio-only, visual-only, early fusion, and late fusion. In the audio modality, both SVM and Conv1D achieved strong results with ComParE_2016 features, with SVM selected for the final system due to its interpretability, simplicity, and stability in high-dimensional spaces. For visual features, BiGRU and CNN+LSTM reached 96% accuracy, and BiGRU was chosen for its robust training behavior and computational efficiency. Late fusion strategies consistently outperformed early fusion by allowing each model to specialize in its modality before merging predictions, avoiding dimensional imbalance, and reducing overfitting risks.

The thesis culminated in the development of *LieBusters,* a complete deception detection prototype capable of real-time webcam and microphone recording, automatic feature extraction, and final prediction using the best-performing models. The system includes a psychologically-aware interface with feedback overlays and audio cues and incorporates LIME-based interpretability to visualize the key factors behind each decision.

Although the results are promising, this study is not without limitations. The small dataset size restricts generalization to wider populations, and the imbalance between feature dimensions across modalities posed challenges, particularly in early fusion. Additionally, the system was evaluated in controlled conditions and may require further optimization for real-world deployment under time and hardware constraints. The text-

based component showed lower performance compared to audio-visual data and may benefit from improved semantic or emotional modeling. Cultural and linguistic diversity was not addressed in this thesis, leaving the question of cross-cultural generalization open for future exploration.

This research opens several avenues for future development, including expanding the dataset for better generalization, enhancing modality synchronization, refining the NLP pipeline, and addressing ethical considerations such as privacy and bias. Ultimately, this thesis bridges the gap between theoretical AI models and practical application by demonstrating the feasibility of intelligent deception detection systems, both in experimental performance and in real-time interactive deployment.

References

[1] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 4th ed., Hoboken, NJ: Pearson, 2020.

[2] E. Kumar, Artificial Intelligence: A Comprehensive Guide, New Delhi: I.K, 2011.

[3] A. M. Turing, "Computing machinery and intelligence," *Mind,* vol. 59, p. 433–460, 1950.

[4] M. L. M. N. R. a. C. S. John McCarthy, "A proposal for the Dartmouth Summer Research Project on Artificial Intelligence," 1955. [Online]. Available: http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html.

[5] E. A. F. a. B. G. Buchanan, "DENDRAL and meta-DENDRAL: Roots of knowledge systems and expert system applications," *Artificial Intelligence,* vol. 59, p. 233–240, 1993.

[6] T. M. Mitchell, *Machine Learning,* McGraw-Hill, Ed., New York, NY, 1997.

[7] OpenAI, "GPT-4 Technical Report," 2023. [Online]. Available: https://openai.com/research/gpt-4.

[8] Y. B. G. H. Yann LeCun, "Deep learning," *Nature,* vol. 521, p. 436–444, 2015.

[9] J. H. M. Daniel Jurafsky, *Speech and Language Processing,* Pearson, Ed., London, 2023.

[10] R. Szeliski, *Computer Vision: Algorithms and Applications,* Springer, Ed., Cham, Switzerland, 2022.

[11] C. M. Bishop, Pattern Recognition and Machine Learning, New York, NY: Springer, 2006.

[12] K. P. Murphy, Machine Learning: A Probabilistic Perspective, 2nd ed., MIT Press, 2022.

[13] E. Algorithms, "Supervised, Unsupervised and Semi-supervised Learning: Key differences," , 5 October 2023. [Online]. Available: https://www.enjoyalgorithms.com/blogs/supervised-unsupervised-and-semisupervised-learning. [Accessed May May 2025].

[14] J. Han, M. Kamber and j. Pei, Data Mining: Concepts and Techniques, 3rd ed., San Francisco, CA: Morgan Kaufmann, 2011.

[15] I. T. Jolliffe, Principal Component Analysis, 2nd ed., New York, NY: Springer, 2002.

[16] A. Makhzani and al, "Adversarial Autoencoders," *arXiv preprint,* 2015.

[17] B. S. A. Z. Olivier Chapelle, *Semi-Supervised Learning,* M. Press, Ed., Cambridge, MA, 2006.

[18] X. Zhu, "Semi-Supervised Learning Literature Survey," 2005.

[19] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed., MIT Press, 2018.

[20] V. Mnih et al, "Human-level control through deep reinforcement learning," *Nature,* vol. 518, no. 7540, p. 529–533, 2015.

[21] A. J. S. a. K.-R. M. Bernhard Schölkopf, "“Kernel principal component analysis”," *International Journal of Pattern Recognition and Artificial Intelligence,* vol. 12, no. 03, p. 265–284, 1998.

[22] N. C. a. J. Shawe-Taylor, "An Introduction to Support Vector Machines and Other Kernel-based Learning Methods," C. U. Press, Ed., Cambridge, 2000.

[23] A. Vidhya, "Support Vector Machines (SVM): A Complete Guide for Beginners," 07 October 2021. [Online]. [Accessed 16 May 2025].

[24] L. Breiman, "“Random Forests”," *Machine Learning,* vol. 45, no. 1, p. 5–32, 2001.

[25] D. W. Hosmer, S. Lemeshow and R. X. Sturdivant, Applied Logistic Regression, 3 rd ed., Wiley, 2013.

[26] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics,* vol. 29, no. 5, p. 1189–1232, 2001.

[27] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, p. 785–794.

[28] J. Brownlee, *Machine Learning Mastery with Python: Understand Your Data, Create Accurate Models, and Work Projects End-To-End,* M. L. Mastery, Ed., 2016.

[29] R. T. J. F. Trevor Hastie, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction,* New York, NY, 2009.

[30] V. M. Sebastian Raschka, *Python Machine Learning,* P. Publishing, Ed., Birmingham, UK, 2019.

[31] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow,* O. Media, Ed., Sebastopol, CA, 2019.

[32] Y. B. a. A. C. I. Goodfellow, Deep Learning, Cambridge: MIT Press, 2016.

[33] I. S. a. G. E. H. K. I. S. a. G. E. H. A. Krizhevsky, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM,* vol. 60, no. 6, p. 84–90, 2017.

[34] M.-W. C. K. L. a. K. T. J. Devlin, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, Minneapolis, 2019.

[35] M. M. a. S. Papert, *Perceptrons: An Introduction to Computational Geometry,* Cambridge, 1969.

[36] G. E. H. a. R. J. W. D. E. Rumelhart, "Learning representations by back-propagating errors," *Nature,* vol. 323, p. 533–536, 1986.

[37] M. S. a. H. W. K. Hornik, "Multilayer feedforward networks are universal approximators," *Neural Networks,* vol. 2, no. 5, p. 359–366, 1989.

[38] L. B. Y. B. a. P. H. Y. LeCun, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, no. 11, p. 2278–2324, 1998.

[39] A. O. S. K. D. E. H. B.-C. J. C. G. H. H. Fabelo, "Hyperspectral Imaging for Glioblastoma Surgery: Improving Tumor Identification Using a Deep Spectral-Spatial Approach," *IEEE Journal of Selected Topics in Quantum Electronics,* vol. 25, no. 1, p. Pages: 1–14, 2019.

[40] S. H. a. J. Schmidhuber, "Long short-term memory," *Neural Computation,* vol. 9, no. 8, p. 1735–1780, 1997.

[41] B. v. M. C. G. D. B. F. B. H. S. a. Y. B. K. Cho, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, 2014.

[42] S. Zhang, Y. Zheng and Y. Qi, "Recurrent Neural Network-Based Language Model for Sentiment Classifi," in *International Conference on Computational Intelligence and Intelligent Systems (CIIS)*, Tokyo, Japan, 2018.

[43] N. S. N. P. J. U. L. J. A. N. G. Ł. K. a. I. P. A. Vaswani, "Attention is all you need," in *Proceedings of the 30th Conference on Neural Information Processing Systems (NeurIPS)*, Long Beach, 2017.

[44] L. B. A. K. D. W. X. Z. T. U. M. D. M. M. G. H. S. G. J. U. a. N. H. A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, Virtual, 2021.

[45] J. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence,* vol. 1, no. 5, p. 206–215, 2019.

[46] L. D. a. D. Yu, "Deep learning: Methods and applications," *Foundations and Trends in Signal Processing,* vol. 7, no. 2014, p. 197–387.

[47] G. Ben-Shakhar and J. J. Furedy, *Theories and Applications in the Detection of Deception: A Psychophysiological and International Perspective,* New York, 2012.

[48] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks,* vol. 61, p. 85–117, 2015.

[49] S. L. King and T. M. Neal, "Applications of AI-Enabled Deception Detection Using Video, Audio, and Physiological Data: A Systematic Review," *IEEE Access,* vol. 12, p. 135207–135240, 2024.

[50] J. Sreerama and G. Krishnamoorthy, "Ethical Considerations in AI: Addressing Bias and Fairness in Machine Learning Models," *Journal of Knowledge Learning and Science Technology,* vol. 1, no. 1, p. 130–138, 2022.

[51] N. R. Council, *The Polygraph and Lie Detection,* T. N. A. Press, Ed., Washington, DC, 2003.

[52] P. Ansley, "Countermeasures in polygraph testing: Practical implications and limitations," *Polygraph,* vol. 33, no. 2, p. 120–132, 2004.

[53] P. Ekman, "Lie catching and microexpressions," in *The Philosophy of Deception*, O. U. Press, Ed., Oxford, 2009, p. 118–133.

[54] P. Ekman and W. V. Friesen, *Facial Action Coding System: A Technique for the Measurement of Facial Movement,* C. P. Press, Ed., Palo Alto, CA, 1978.

[55] Z. Zeng, M. Pantic, G. I. Roisman and T. S. Huang, "A survey of affect recognition methods: Audio, visual, and spontaneous expressions," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 31, no. 1, p. 39–58, 2009.

[56] H. Hollien, *The Acoustics of Crime: The New Science of Forensic Phonetics,* Springer, Ed., New York, 2002.

[57] J. L. Patrick and C. J. Iacono, "Voice stress analysis: A review of evidence," *Journal of Forensic Sciences,* vol. 48, no. 4, p. 740–753, 2003.

[58] N. R. Council, *The Polygraph and Lie Detection,* T. N. A. Press, Ed., Washington, DC, 2003.

[59] S. V. R. K. M. &. S. A. K. Sondhi, "Voice analysis for detection of deception," in *2016 11th International Conference on Knowledge, Information and Creativity Support Systems (KICSS)*, Thailand, 2016.

[60] A. Vrij, "Detecting lies through nonverbal behavior: Theoretical, empirical, and practical perspectives," in *The Detection of Deception in Forensic Contexts*, C. U. Press, Ed., Cambridge, 2010, p. 15–34.

[61] B. M. DePaulo, J. J. Lindsay, B. E. Malone, L. Muhlenbruck, K. Charlton and H. Cooper, "Cues to deception," *Psychological Bulletin,* vol. 129, no. 1, p. 74–118, 2003.

[62] C. F. J. Bond and B. M. DePaulo, "Accuracy of deception judgments," *Personality and Social Psychology Review,* vol. 10, no. 3, p. 214–234, 2006.

[63] L. Breiman, "Random forests,," *Machine Learning,* vol. 45, no. 1, p. 5–32, 2001.

[64] V. Pérez-Rosas, R. Mihalcea and M. Burzo, "Deception detection using real-life trial data," in *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction (ICMI)*, Seattle, WA, USA, 2015.

[65] J. K. Burgoon, J. G. Proudfoot, R. M. Schuetzler and D. Wilson, "Predicting Deception in Real Time Using Automated and Manual Coding of Nonverbal Behavior," *IEEE Transactions on Affective Computing,* vol. 6, no. 4, p. 337–350, 2015.

[66] O. Abdel-Hamid, A. Mohamed, H. Jiang and G. Penn, "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing,* vol. 21, no. 7, p. 1530–1541, 2013.

[67] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," in *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, 2005.

[68] J. Chung, C. Gulcehre, K. Cho and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555,* 2014.

[69] D. Tran, L. Bourdev, R. Fergus, L. Torresani and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[70] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit and N. Houlsby, " An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations (ICLR)*, 2021.

[71] S. Yan, Y. Xiong and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[72] D. e. al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Minneapolis, USA, 2019.

[73] M. Bahaa, M. Hany and E. E. Zakaria, "Advancing Automated Deception Detection: A Multimodal Approach to Feature Extraction and Analysis," *arXiv preprint arXiv:2407.06005,* 2024.

[74] V. Pérez-Rosas, M. Abouelenien, R. Mihalcea and M. Burzo, "Deception Detection using Real-life Trial Data," in *Proc. ACM Int. Conf*, Multimodal Interaction, 2015, p. 59–66.

[75] M. F. Chang, J. Whitaker and J. Z. Xu, "Verbal Deception Detection Using Fine-Tuned Language Models," *Scientific Reports,* vol. 13, no. 1, p. 1–12, 2023.

[76] A. N. Lang and M. Stricker, "Deception Detection Using Eye Movements and Machine Learning Models," *arXiv preprint,* 2024.

[77] L. Floridi and e. al, "AI4People—An Ethical Framework for a Good AI Society: Opportunities, Risks, Principles, and Recommendations," *Minds and Machines,* vol. 28, no. 4, p. 689–707, 2018.

[78] V. Gupta, M. Agarwal, M. Arora, T. Chakraborty, R. Singh and M. Vatsa, "Bag-of-Lies: A Multimodal Dataset for Deception Detection," *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW),* 2019.

[79] K. Radlak, M. Bożek and B. Smolka, "Silesian Deception Database: Presentation and Analysis. ResearchGate," 2016.

[80] S. K. Elkins, S. Aryal and J. Whitehill, "Celeb-Lie: A Multimodal Dataset for Deception Detection in Celebrity Interviews," in *arXiv preprint*, 2023, p. 2301.06571.

[81] J. Hirschberg, R. Levitan and S. I. Levitan, "Deception Detection via Prosodic, Lexical and Acoustic Features Using Machine Learning," in *INTERSPEECH 2015*, Conference of the International Speech Communication Association, 2015, p. 1476–1480.

[82] S. L. X. L. K. Z. Z. W. J. T. H. X. J. C. Y. M. Z. C. e. a. Cong Cai, "MDPE: A Multimodal Deception Dataset with Personality and Emotional Characteristics,," *arXiv preprint,* p. 2407.12274, 2024.

[83] F. Eyben, F. Weninger, F. Gross and B. Schuller, "Recent Developments in openSMILE, the Munich Open-Source Multimedia Feature Extractor," in *Proceedings of the ACM International Conference on Multimedia*, 2013.

[84] B. Logan, "Mel Frequency Cepstral Coefficients for Music Modeling," in *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, 2000.

[85] T. M. Ribeiro, S. Singh and C. Guestrin, "Why Should I Trust You?," *Explaining the Predictions of Any Classifier,* p. 1135–1144, 2016.

[86] z. Mohamed et al, "IEEETrans2022.pdf," 2022. [Online]. Available: https://public.websites.umich.edu/~zmohamed/PDFs/IEEETrans2022.pdf.

Appendix: Access to Source Code

The final version of the real-time deception detection system developed in this study — **Lie Busters** — is available on GitHub:

GitHub Repository:
https://github.com/MERx21/LieBusters.git

This repository includes:

- The complete Lie Busters web application
- Audio and visual feature extraction scripts
- Pre-trained model loading and prediction code
- Flask backend and horror-themed user interface
- Example files and runtime instructions

Please note that this repository contains only the final deployed system. Model training notebooks and experimental results are not included.

All materials are shared for academic and demonstration purposes.

To complement the code, a video demonstration of **Lie Busters** may be uploaded to the following YouTube channel:

YouTube Channel:
(96) meram tadjine - YouTube

The video (if available) will showcase:

- How the system records and analyzes audio-visual data
- The real-time interface and deception prediction process
- Key system functionalities and a brief overview

This is intended to help readers visually understand the system's operation and practical use.