Multimodal Deception Detection Using Linguistic and Acoustic Features

Tien Nguyen, Faranak Abri, Akbar Siami Namin, and Keith S. Jones

Abstract Recently, there has been a growing interest among researchers in the automatic detection of deceptive behavior, actions, and contents. This surge in attention is driven by the wide-ranging applications of deception detection, particularly in criminology and cybersecurity. To advance this line of research, this study investigates both text and audio data derived from speeches in natural languages. We evaluate traditional linguistic models alongside deep models and advanced Large Language Models (LLMs), utilizing Natural Language Processing (NLP) techniques to model deception detection. Furthermore, we employ various feature selection methods to determine the significance of linguistic features. Through extensive experimentation, we assess the effectiveness of both conventional and advanced deep models on transcribed data while also applying deep models to audio data, thus leveraging both types of data to build a multimodal model for deception and lie detection. Our findings indicate that the Bidirectional Long Short-Term Memory (BiLSTM) model excels in processing textual data. On the other hand, the ResNet50 model performs best with audio data. By combining these models in a late fusion approach, we achieve a model that outperforms individual text and audio models.

T. Nguyen · F. Abri (✉)
San José State University, San Jose, CA, USA
e-mail: faranak.abri@sjsu.edu

T. Nguyen
e-mail: tien.t.nguyen04@sjsu.edu

A. Siami Namin · K. S. Jones
Texas Tech University, Lubbock, TX, USA
e-mail: akbar.namin@ttu.edu

K. S. Jones
e-mail: keith.s.jones@ttu.edu

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2025
M. Stamp and M. Jureček (eds.), *Machine Learning, Deep Learning and AI for Cybersecurity*, https://doi.org/10.1007/978-3-031-83157-7_20

1 Introduction

Attackers have always used deception to manipulate or take advantage of the trust of others for their own benefit. It can be used to scam people out of their money or mislead a criminal investigation. This deceptive behavior can lead to serious consequences, such as innocent people being wrongly convicted or losing money. With that motivation in mind, finding out when someone is not telling the truth could help avoid damage in both interpersonal and work relationships. Developing effective techniques to know when people are deceiving is crucial to supporting fair judgment in court cases. The ability to recognize dishonest claims is a solution to the dilemma our society faces every day. Applications of such deception detection are not limited to legal and criminal contexts but can extend to areas such as financial fraud prevention, corporate security, and safeguarding personal interactions where trust is crucial. Additionally, deception detection from speech can be particularly valuable in cybersecurity, such as identifying phishing calls and other forms of verbal deception.

Traditionally, authorities have performed polygraph tests on suspect individuals [18]. This test uses a device to measure and record physiological indicators such as heartbeat, blood pressure, and skin conductivity, based on the assumption that deceptive answers trigger involuntary physiological reactions [18]. Although the polygraph has long been considered a standard method for detecting lies, experimental studies show that these tests are not always accurate. They are prone to errors and biases that arise from both the equipment used and human misinterpretation [10]. In addition, many factors can affect physiological responses, which may not necessarily indicate deception. An alternative approach, as proposed in this work, is the use of multimodal lie detection techniques that integrate multiple data sources, such as speech and facial expressions, to enhance the accuracy of lie detection tasks and address the limitations of traditional polygraph tests. Specifically, analyzing speech data can provide a wealth of information to identify instances of deceitfulness [4, 15]. One potential approach is to examine the characteristics in the audio of the speech, often referred to as "audio features", also known as Mel-frequency cepstral coefficients (MFCC), such as changes in pitch, speaking rate, and intensity, which can occur when someone is untruthful [4, 28, 31]. Another potential approach involves analyzing transcribed data of speech, which is often referred to as "textual or transcribed features", such as word choice and sentence structure, which can also serve as indicators of deception [6, 14, 24]. In addition, considering nonverbal cues, often referred to as "behavioral features", such as expressions, body language, and eye movements, is another feasible approach to detecting deceptive behavior [6, 31].

The recent advancement in machine learning and deep learning algorithms enables the creation of classification schemes trained on these multimodal features to accurately classify people's truthfulness in a given case or scenario. The use of machine learning in lie detection has gained significant attention in recent years, with many researchers achieving promising results in accurately detecting deception from speech by combining these features and using machine learning algorithms [3, 27].

This work aims to create a multimodal model that uses both transcribed text and audio/speech data to detect deceptive behavior more accurately. Recognizing the potential of multimodal approaches, we research and conduct experiments to develop models that can better identify deception. By focusing on both linguistic and audio cues, our objective is to reduce the risk of wrongful convictions and financial fraud, ultimately improving the fairness and reliability of legal and investigative processes. One of our key contributions is the integration of text and audio models using a late fusion technique. This approach takes advantage of the strengths of both modalities, resulting in a more robust detection system. We also test our models on the Real-Life Trial dataset to ensure that our findings are grounded in practical, real-world scenarios. By experimenting with conventional models, LLMs, and deep learning models, we provide a comprehensive and reliable method for deception detection.

This paper is structured as follows. Section 2 reviews existing research in the field of deception detection. Section 3 presents the technical background of the conventional and deep models we studied. Section 4 outlines the experimental setup, giving details on the dataset, preprocessing, feature selection, and evaluation metrics. Section 5 discusses the setup of the detection models, and Sect. 6 explains the performance of these models. Section 7 discusses these results, providing insight into the effectiveness and alignment of various approaches with existing literature. Lastly, Sect. 8 concludes the paper and suggests future work.

2 Related Work

This section will review the datasets and models used and implemented in previous research on deception detection. A summary of the reviewed papers is provided in Table 1.

2.1 Existing Public Datasets

In the process of training models that are capable of detecting deception, a variety of datasets are utilized, such as Real-life trial [4], Columbia X-Cultural Deception [20], or H Wolf [8] dataset. These datasets are typically sourced from different contexts and scenarios, providing a wide range of data from which models can learn. Each dataset can fall into three main categories: (1) The first category is any data collected from real-life situations where deception is common, such as legal proceedings; (2) Data can also be generated artificially by asking people questions designed to force deception; and (3) Finally, data can also be collected while playing games, where deception can also commonly and naturally occur.

The first category of data comes from various real-life situations. This type of dataset provides authentic examples of deception and truth-telling. Instead of staged setups, these datasets can provide more realistic scenarios for lie detection. Some research papers [4, 13, 23, 24] have used data collected from actual court trials. The dataset includes 121 videos divided into 61 clips that show deceptive behavior

Table 1 A comparison of the performance of existing work

Ref.	Dataset	Features	Modality	Classification method	Metrics	Scores (%)
[22]	Nearly 1,500 true and false statements	43 features	Transcription, written	SVM	Accuracy	58.9
[20]	Columbia X-cultural deception	Ngrams, acoustic-prosodic including pitch, intensity, duration, voice quality (jitter, shimmer, and harmonics-to-noise ratio), spectral harmonicity, and psychoacoustic spectral sharpness, MFCCs	Audio, transcription	Hybrid: LSTM + DNN	Precision recall F1	67.32
						60.80
						63.90
[15]	Political debates	LIWC, TF.IDF, i-vector, speaker psychoacoustic spectral sharpness, voice quality, energy, spectral, and other low-level descriptors	Audio, transcription	Logistic regression, multi-input feed-forward neural network	Accuracy MAE MMA MA F1 MAR	67
						69
						51.04
						45.07
						47.25
[4]	Real-life trial	MFCC and mean of MFCC of all frames	Audio	SVM using polynomial kernel	Accuracy	81
[8]	H Wolf	Acoustic statistics and time-frequency features	Audio	Hybrid semi-supervised neural network	Accuracy	68.62
[28]	IDIAP WOLF	Fundamental frequency, short-term energy, zero-crossing rate, MFCC, other features	Video	SVM	Accuracy	82.47

(continued)

Table 1 (continued)

Ref.	Dataset	Features	Modality	Classification method	Metrics	Scores (%)
[13]	Real-life trial	136-dimension vectors	Visual, audio, transcription	BiLSTM, ensemble model (voting)	Accuracy	96
[24]	Real-life trial	Unigrams, LIWC, facial displays, hand gestures, facial action units (FACS), pitch, silence and speech histograms	Visual, audio, transcription	NN	Accuracy	84.18
[31]	Real-life trial	Ngrams, audio subsystem: MFCCs, log-energy of MFCC, and CC, micro-expressions	visual, audio, transcription	SRKDA (audio) + SVM (text) + AdaBoost (micro-expression)	Accuracy	97
[14]	Blind tasting game	LIWC, binary and numeric features capturing hedging linguistic and syntactical distinctiveness, acoustic-prosodic including intensity mean and max, pitch mean and max, 3 voice quality features (shimmer, jitter, noise-to-harmonics ratio), and speaking quality, MFCCs	Visual, audio, transcription	MLP + BiLSTM + Multi-task	F1	71.51
[6]	Real-life trial	Pitch-based features, 21 text-based features: the use of passive voice, negative emotion words, deny words, hasYes, laughing, crying, as well as speech errors, etc.	Visual, audio, transcription	Hybrid: KNN	Accuracy	97
[33]	Real-life trial	21 pitch-based and spectral features, BiLSTM text features, ResNet video features	Visual, audio, transcription	GCFM + early fusion (cross model attention)	Accuracy F1	88.14
						78.5
[23]	Real-life trial	Mel Spectrograms features, word dimension vector features, video frame dimension features	Audio, transcription video	LSTM + BiLSTM + CNN + RestNet50	Accuracy	96

and 60 clips that feature truthful interactions [24]. The individuals featured in these videos, who are either defendants or witnesses, range in age from 16 to 60. On average, the videos are approximately 28 s long, and the transcripts of these videos have an average of around 66 words, amounting to a total of over 8,000 words of speech data. Another notable dataset is used by Kopev et al. [15]. This dataset is for real-world political debate and offers a wider range of realistic situations for lie detection, including claims labeled as true, half-true, or false. This dataset consists of 94 training claims and 192 test claims, providing substantial data for the models to learn from.

The second type of dataset is generated by asking actors questions to generate false and true responses. In a study by Sarzynska-Wawer et al. [22], 400 participants were invited to create four statements on a given topic: two of these statements were to be delivered orally, while the other two were to be written. This method resulted in 1,600 statements, 1,498 of which were selected for the final analysis. Mendels et al. [20] used the CXD corpus, which comprises deceptive and non-deceptive speech from native English and Mandarin speakers, all communicating in English. It includes 170 conversations involving 340 participants. This data was gathered using a fake resume setup where subjects alternated between interviewer and interviewee roles, answering 24 biographical questions. Participants had financial incentives to both lie convincingly and accurately detect lies. During the interviews, the interviewees labeled each response as true or false.

The third type of dataset was collected from playing games. Tao et al. [28] used the IDIAP WOLF dataset developed by the Swiss IDIAP Research Institute. This study collected vocal signals from the "werewolf killing game" that involved 12 participants, four of whom played werewolves to create confusion through deception while the rest of the players played honest characters. The werewolves are expected to lie, while the other players need to guess who the werewolves are. Similarly, Fu et al. [8] created the H-Wolf corpus, a self-built dataset constructed from the Idiap Wolf and Killer datasets. They gathered approximately 70 h of video from the "Werewolves of Miller's Hollow" competitions available online, selecting clips that contained truthful and deceptive interactions based on the players' ID cards and the rules of each game.

2.2 Conventional Models in Deception Detection

This section will review some of the conventional models used in previous work for deception detection. Researchers have explored a variety of methodologies, ranging from traditional statistical models to advanced machine-learning techniques for this problem. Sarzynska-Wawer et al. [22] implemented a Support Vector Machine (SVM) and XGBoost model with 20-fold cross-validation. Their best model, SVM, gave an accuracy of 58.9%.

Bareeda et al. [4] built SVM-based classifiers using Gaussian and polynomial kernels. Based on their experiments, they found that using polynomial or Gaussian kernels resulted in an overall classification accuracy of 81 and 78% for the lie and truth classes, respectively.

Tao et al. [28] extracted different acoustic features from audio datasets to detect deception using SVM as the classifier. The experimental results showed that SVM could effectively detect deception with an accuracy of over 80%.

Chebbi et al. [6] created K Nearest Neighbour (KNN) models for each modality (visual, audio, transcription) separately using feature selection techniques to select the most relevant features. They combined the modalities using a decision-level fusion approach based on belief theory. The approach was studied using the real-life trial dataset [24]. The deception detection accuracy rate reached 97% using only 19 combined features.

Şen et al. [24] collected videos from a set of actual court trials and built models that used verbal, acoustic, and visual modalities to detect deception. Initially, they conducted experiments with each set of features separately using SVM, Randon Forest (RF), and Neural Network (NN) classifiers. Then, they tried various combinations of features using early and late fusion. Their results showed that late fusion achieved better performance with 84.18% accuracy with combined text, visual, and acoustic features.

Venkatesh et al. [31] introduced a novel deception detection approach that used different types of data, including audio, text, and nonverbal characteristics, to build their deception detection models. The method combined the results of each of these features using majority voting. Specifically, the audio component was based on Cepstral Coefficients (CC) and Spectral Regression Kernel Discriminant Analysis (SRKDA). On the other hand, the text model used bag-of-n-gram features and a linear SVM classifier, whereas the nonverbal component employed the AdaBoost classifier. The results showed that the proposed method outperformed both existing state-of-the-art techniques and human performance, achieving a deception detection accuracy of 97% in the entire dataset during a 25-fold cross-validation.

2.3 Deep Learning Models for Deception Detection

Deep learning models have significantly impacted deception detection, resulting in a level of complexity that can detect certain details in the data. This section reviews previous work using deep models. Sehrawat et al. [23] proposed a model that combined Long-Short Term Memory (LSTM), Bidirectional Long-Short Term Memory (BiLSTM) networks, Convolutional Neural Network (CNN), and ResNet50 to detect deception. They first extracted text, audio, and video features from the "Real Life Court Trial" dataset. To process audio data, they transformed them into Mel spectrograms to create a visual representation that captured key audio characteristics. ResNet50 was then used to analyze these Mel Spectrograms. The proposed model, which used audio and text features, achieved an accuracy of 80%.

Unlike Sehrawat et al. [23], who used an existing dataset, Marcolla et al. [19] created their own dataset by interviewing subjects to capture the subject's answers, labeled lying or truthful. To get the audio features, the researchers used Librosa library functions to extract the Mel-Frequency Cepstral Coefficients (MFCC) characteristics. The researchers then normalized the features through the padding to match the length of the longest sequence, ensuring uniformity for neural network processing [19]. Their LSTM neural network model resulted in an overall classification accuracy of 72.5%. Hsiao and Sun [13] also used MFCC for their audio feature. But instead of normalizing MFCC features to match the longest sequence, they calculated average MFCC values per second. This method reduced MFCC length, which helped train their BiLSTM model more effectively. They also extracted features from text and transcript. Lastly, they proposed an ensemble model that combined the outputs of the audio, visual, and transcription models using BiLSTM. Their ensemble model achieved 96% accuracy when used on the "Real Life Court Trial" dataset.

Gallardo-Antoln and Montero [9] developed an automatic deception detection model based on gaze and speech characteristics using attention-based LSTM. The feature extraction procedure from gaze data involved selecting channels from the Gazepoint GP3 Eye Tracker for fixations, saccades, and pupil size, which are known as indicators of deceptive behavior. For speech, features were derived from Log-Mel Spectrograms using Python's package LibROSA. The researchers trained their models on the Bag-of-Lies dataset and achieved an accuracy of 70.5%.

Zhang et al. [33] created a Graph-based Cross-modal Fusion Model (GCFM) along with a Cross-modal Attention Mechanism to detect deception in the Real-Life Trial dataset [24]. They extracted visual, textual, and audio features by using a pre-trained ResNet50 and LSTM neural network with attention mechanisms. The proposed GCFM method achieved an accuracy of 88.14% as well as an F1-score of 78.50%. Additionally, using association learning increased the accuracy by 1.87% while the cross-modal attention mechanism improved the accuracy by 2.44%.

3 Background on Conventional and Deep Models Studied

3.1 Conventional Models

We explore Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Logistic Regression (LG) for deception detection using textual data. SVM is a supervised learning algorithm commonly used for classification tasks. SVM aims to identify a hyperplane in an n-dimensional space, where n represents the number of features, to effectively separate data points into different classes [12]. KNN is a non-parametric algorithm that stores all training data and then classifies new data points based on the "k" closest training points, where k is some constant number of points. LG is a statistical model commonly used for classification tasks. It outputs probabilities from 0 to 1 for different classes and classifies data based on continuous and discrete measurements. The model finds a line or a hyperplane in higher-dimensional spaces that best separates the data into classes. After trying various lines, the one with the maximum likelihood is selected.

3.2 Convolutional Neural Networks (CNNs)

CNNs are powerful tools in deep learning, particularly for analyzing visual and image data. The CNNs architecture, as shown in Fig. 1, contains many layers that perform different tasks to convert input data into important features. The first layer, called the convolutional layer, uses many filters on the input image. The filter output is put on a feature map (Fig. 2). After that, the feature maps are run through an activation function such as ReLU. Next, these feature maps enter the pooling layers. The purpose is to decrease the spatial dimensions of the data while keeping important information intact. The pooled layers are converted to columns of the input nodes. Finally, fully connected layers take the input nodes and compute the final classification task. CNN's architecture for visual recognition tasks is powerful because it can learn complex features hierarchically. This method demonstrates impressive accuracy in image classification, object detection, and other fields of application.

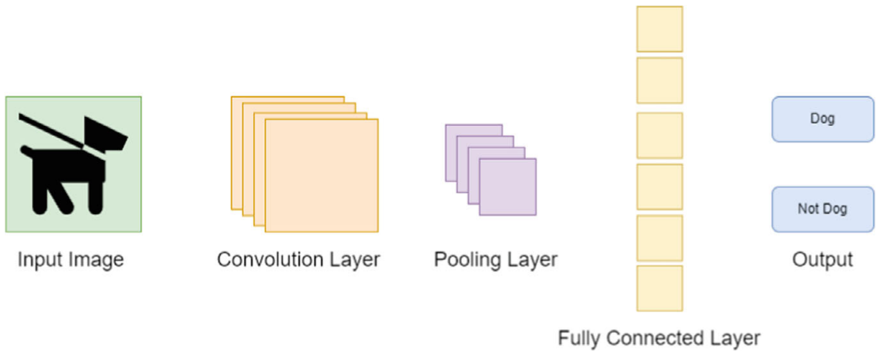

Fig. 1 A simple architecture of CNNs

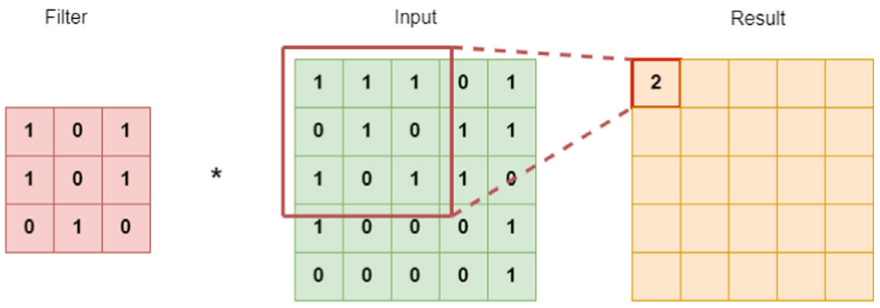

Fig. 2 Process of applying filter on input image and the results onto feature map

3.3 Long Short-Term Memory (LSTM)

Recurrent Neural Networks (RNNs) are neural networks that process data sequences such as time series or natural language. They have looped connections that let them keep information over time. This makes RNNs good for tasks needing context from earlier data. But RNNs have a big problem: vanishing or exploding gradients. During training, the gradients used to update weights can get too small (vanishing) or too big (exploding). This makes it hard for RNNs to learn long-term patterns. The LSTM is an advanced RNN that can solve RNNs' vanishing/exploding gradient problem. The memory cell of the LSTM network contains three different gates: the input gate, the forget gate, and the output gate. The input gate determines what information we should store in the memory cell, while the forget gate chooses which information to remove from the memory cell [30]. The function of the forget gate is to manage and control what is output from the memory cell. Figure 3 describes the architecture of an LSTM block.

3.4 Bidirectional LSTM

The BiLSTM model leverages complete sequential information by considering both past and future data points for each position in the sequence, thereby enhancing the original LSTM designed for sequence learning [32]. BiLSTM consists of two LSTMs, and both of them return a probability vector. Their combination forms the final output. This ability of BiLSTMs to process in both directions is especially useful for intricate sequence prediction tasks such as examining speech and text. Figure 4 describes the architecture of a BiLSTM block.

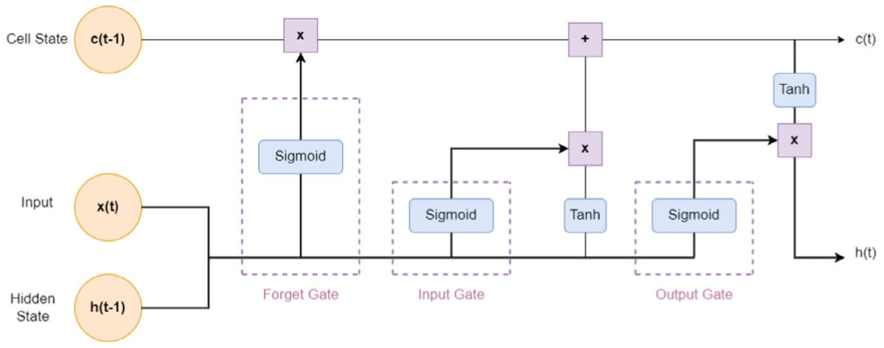

Fig. 3 Architecture of an LSTM block. Adapted from [17]

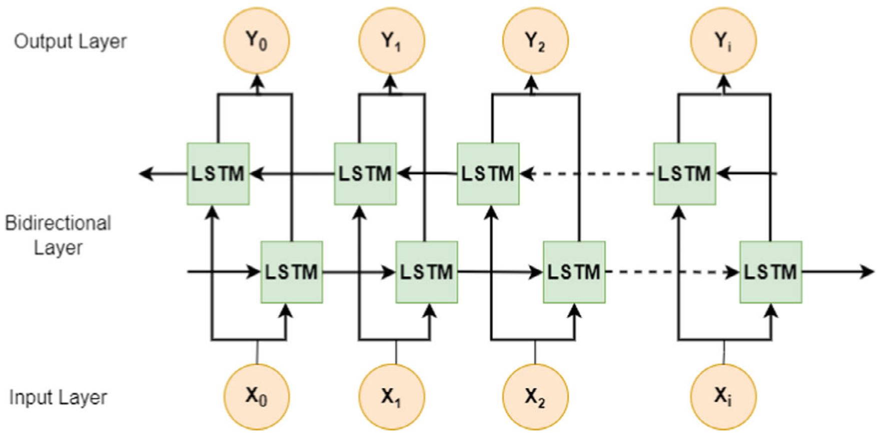

Fig. 4 Architecture of a BiLSTM block. Adapted from [5]

3.5 Residual Network 50 (ResNet50)

ResNet50 is a convolutional neural network (CNN) with 50 layers. ResNet50 is useful for complex tasks that involve image processing and analysis. It contains 48 convolution layers, one MaxPooling layer, and one AveragePooling layer. The main idea behind the ResNet50 model is its unique design using residual blocks. These residual blocks help to solve the vanishing gradient problem. The blocks with residuals have "skip" connections. This lets the layers learn residual mappings, which means that the network can understand these mappings instead of only direct feature mappings [11]. This way of building makes it possible to create deeper networks without losing performance and helps to improve the flow of gradients during training, making learning more efficient and stable. This proven CNN is especially useful for feature extraction in complex datasets, including audio spectrograms. Figure 5 describes the architecture of a RetNes50 block.

3.6 Late Fusion

We also apply a late fusion method, combining audio and text data characteristics. Each data type is processed using its own specialized neural network architecture. The results of both models are combined into one vector after being processed separately. Afterward, this combined vector is fed into a fusion layer. Figure 6 describes a high level of late fusion model. In this layer, the weights that can be trained are used to find the best weight for each model. The weights have a softmax function applied to them. Softmax will force all outputs to sum up to one. The output of this layer is then processed by a last dense layer with sigmoid activation. Each output provides a probability score showing the possibility of deception. This method, called late

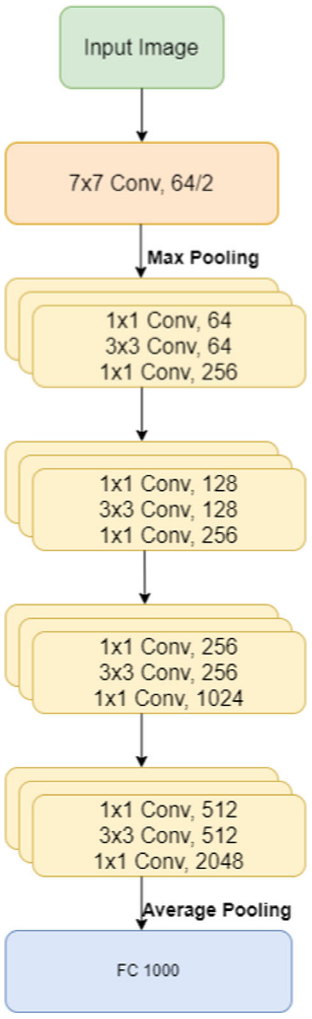

Fig. 5 Architecture of a RetNes50. Adapted from [23]

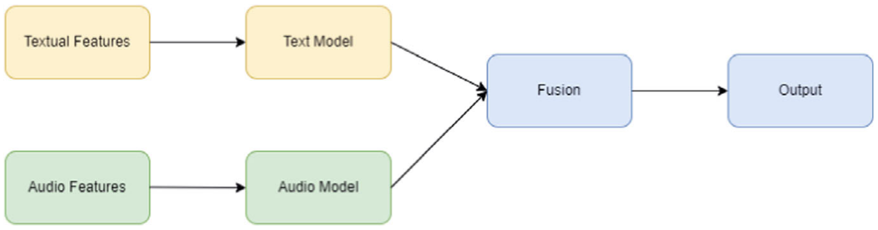

Fig. 6 A high level of late fusion model

fusion, makes our model flexible and adjustable; it learns which features from every data type are more telling about deceitful behavior.

3.7 Additional Pretrained Models

In addition to the models that are fine-tuned throughout the research, several pre-trained models are applied for comparative analysis. We choose to use pre-trained models because they offer a strong starting point, having already learned from large datasets. These models are known for capturing complex patterns in text and audio data, which are crucial for deception detection. Using these models, we aim to benefit from their proven effectiveness in various tasks and expect them to enhance the accuracy of our detection system. These include BERT, a language model developed by Google that uses a bidirectional approach and transformer architecture to create context-aware word representations, enabling strong performance in various natural language processing [7]. We also use GPT-2, developed by OpenAI, which generates coherent and contextually relevant text based on extensive pre-training on large datasets [26]. Furthermore, RoBERTa, an improved version of BERT developed by Facebook AI Research, optimizes the pretraining process through hyperparameter adjustments and other enhancements [16]. Lastly, the VGG-16 model, a convolutional neural network with 16 weight layers, is utilized for its effectiveness in image classification tasks, focusing on 3×3 convolution layers and a simplified architecture [29].

4 Experimental Setup

4.1 Methodology

We conducted our experiment as follows: We started with conventional models that use linguistic features extracted from the textual data, then progressed to deep models such as BiLSTM and pre-trained models that we covered in our previous work [21]. These text-only experiments yield relatively good results, indicating that deep models work well. In this work, we add audio data using a similar analysis. We use deep models like ResNet50 and VGG16 to analyze speech Mel spectrograms. Finally, a late fusion technique is employed to combine the outputs of the best-performing textual and audio models. The details of all the steps completed prior to and included in this work are outlined in this section.

4.2 Dataset Description

Our research uses a unique and valuable dataset from public court trials created by researchers at the University of Michigan [24]. This dataset contains 121 video

Table 2 Example of deceptive and truthful content

Deceptive
And he told me that, ammm ... he was trying to figure some stuff out, and ammm ... I asked him Like what? and he will ... I mean I will never forget it, he was smoking a cigarette, and he was like really calm, and he looked at me and he said What would you say if I said ... if I told you Laura was dead? And I was like, you know, I was like What? And ... basically he told me that, ammm ... the night that Laura had come over to the house, that she had died, and that whenever I left that he just panicked and freaked out, and I got ... I started freaking out, and I was asking him why he didn't call the cops [stutters] ... call for help like he told me he was going to, and he told me that, ammm ... he got scared that he was a black man with a dead white woman and nobody was gonna believe him that it was an accident
Truthful
I have no idea. A police officer I presume. You'd have to ask my mother or my brother. Nope. They said they didn't know where he was being taken. Yep. Went to the house, I was in a fairly catatonic state, my dad and my brother started making phone calls to all the local hospitals, and they eventually got a hold of... I don't know, whatever the hospital is, Atlanta Medical Center. And they wouldn't tell my dad anything but that he was being taken there. So we got in the car, and we left. That's correct. Yes, he was and I had, I – that was instructed that that was the best idea was to keep him at the day care. The, uh ... Donna. The woman that runs the day care. Yep. That's the safest place ... uh for him to be

recordings, evenly split between deceptive and truthful statements. Each video is about 28 s long and features defendants or witnesses speaking in different trial scenarios. The dataset includes not only the video content but also transcripts of each video, allowing us to analyze textual information alongside audio/visual cues. Additionally, the research group has annotated each video with gestures such as smiles, laughs, etc. In this work, we primarily focus on the textual and audio data for building a multimodal deception detection model. Table 2 provides examples of both deceptive and truthful content from the dataset.

4.3 Transcribed Data

4.3.1 Preprocessing and Cleaning

Our text processing pipeline varies depending on the model. For conventional models, we remove non-alphabetic characters from the text to eliminate noise that could interfere with later processing stages. This step ensures that only alphabetic letters remain, which helps the models focus on relevant linguistic features. We accomplish this using Python packages such as re for regular expression operations and NLTK for text processing tasks. The cleaned text is then used to extract features, which will be discussed in the next section.

In contrast, for deep models, we do not remove non-alphabetic characters, allowing the model to retain as much of the original text as possible. We apply stemming to each word, reducing it to its root form by removing suffixes, which helps reduce the number of unique words for more efficient analysis. After stemming, we perform one-hot encoding, where each word is assigned a unique integer index from a vocabulary size of 5000. Additionally, each sequence is padded with zeros to a uniform length of 221 words, which is the maximum sentence length in our dataset. (221). Padding is necessary because machine learning algorithms require input data with a consistent shape. It ensures all sequences have the same feature count, which helps the machine learning algorithms function correctly.

4.3.2 Linguistic Features Extraction

After preprocessing textual data, we extract 16 key textual features relevant to lie detection, detailed in Table 3. Specifically, to understand the speaker's perspective, discourse structure, and temporal references, we compute the number of pronouns, conjunctions, and verb tenses (i.e., past, present, future). The sentiment score aggregates the emotional valence of words into a compound score ranging from -1 to 1, indicating negative, neutral, or positive emotion. We also use part-of-speech tagging to get the frequency of adjectives and adverbs, which helps provide insights into the descriptive language used. Additionally, we count the frequency of filler words like 'um,' 'uh,' and 'like,' as well as repetitions, negations, and self-references to evaluate the speaker's fluency, rhetorical style, and persuasive attempts. Together, these features provide a comprehensive framework for detecting deception through linguistic patterns in textual data.

Table 3 Description for extracted linguistic features [21]

Feature name	Description
Word count	The total number of words in the text
Sentence count	The total number of sentences in the text
Sentiment score	A numerical score indicating the overall sentiment of the text
Average word length	The average length of words in the text
Vocabulary diversity	The ratio of unique words to the total number of words in the text
Adjective frequency	The proportion of adjectives in the text
Adverb frequency	The proportion of adverbs in the text
Pronoun frequency	The proportion of pronouns in the text
Conjunction frequency	The proportion of conjunctions in the text
Past tense frequency	The proportion of verbs in the past tense in the text
Present tense frequency	The proportion of verbs in the present tense in the text
Future tense frequency	The proportion of verbs in the future tense in the text
Filler word count	The number of common filler words in the text
Repetition count	The proportion of words that appear more than once in the text
Negation count	The number of negations in the text
Self-reference count	The number of self-referential words in the text (e.g., "I," "me," "myself")

4.3.3 Feature Selection for Conventional Models

We use two main methods to choose important features from the textual data: (1) overlapping coefficient (OVL) and (2) stepwise regression. The OVL method assesses the significance of specific features by comparing the probability density functions (PDFs) of features between "Lie" and "Truth" categories [1]. Features with lower OVL scores, indicating less overlap, are deemed more effective for distinguishing between these categories. On the other hand, higher OVL scores suggest more overlap, which implies that the feature is less effective at distinguishing between the categories because the distributions are more similar. The stepwise regression method, a greedy approach, iteratively adds or removes features based on their impact on model performance [2]. Typically, this evaluation involves training the model with the selected features and measuring its performance using accuracy or F1 score metrics. Based on these performance metrics, a decision is made to include or exclude a feature, and the process continues until a predefined stopping condition is met.

4.4 Audio Data

We convert MP4 video files into WAV audio format using the movies library. Our approach is to transform the audio into images of the type of Mel spectrograms using the Librosa library. We use a Mel spectrogram, a visual representation of sound that aligns frequencies to the Mel scale (corresponding to human auditory perception). This is achieved by segmenting the audio, performing a Fourier transform on each segment to identify frequency content, and then applying Mel scale filters to emphasize perceptually important frequencies. Finally, the Mel spectrograms are converted into the RGB color space using matplotlib library and resized to a uniform dimension of 224×224 pixels to ensure consistent input size. Once the Mel spectrogram images are generated, we use them as input data for training deep learning models. Examples of deception and truth images are shown in Figs. 7 and 8. The Mel spectrogram's x-axis represents time in seconds, while the y-axis shows frequency in Hertz on the Mel scale, which emphasizes frequencies important to human hearing. The color scheme indicates sound intensity, with lighter colors representing louder sounds and darker colors indicating quieter ones. The color bar, in decibels (dB), provides a reference for these intensity levels.

4.5 Model Evaluation

We evaluate the models using 5-fold cross-validation. It divides the dataset into five subsets and iteratively creates the training and test sets. During each iteration, one subset is used as the test set, while the remaining four subsets form the training set. This process is repeated five times, with each subset taking a turn as the test set. This

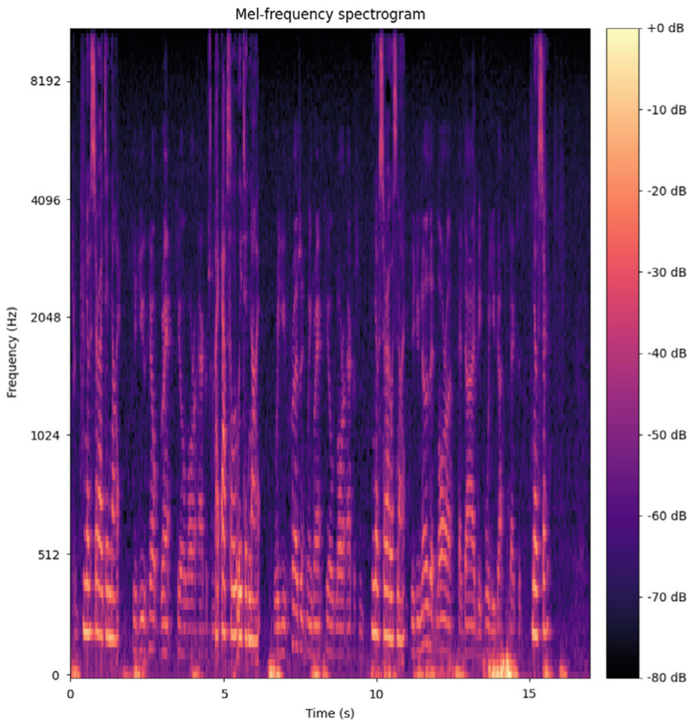

Fig. 7 Deception image example

ensures robustness and minimizes overfitting. The model's performance is evaluated using metrics such as accuracy and the F1 score, where the following metrics are used for the computation:

- TP: true positives (classifier correct; classifier guessed 1).
- FP: false positives (classifier incorrect; classifier guessed 1).
- TN: true negative (classifier correct; classifier guessed 0).
- FN: false negative (classifier incorrect; classifier guessed 0).
- Accuracy measures the percentage of correct predictions out of the total instances.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

- The F1 score is the harmonic mean of the precision and recall metrics. Precision measures the percentage of times the classifier was correct when it was predicting the true (1) class. Recall is the percentage of times that the model correctly predicted 1 when the label was, in fact, 1.

$$\text{Recall} = \frac{TP}{TP + FN} \qquad \text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

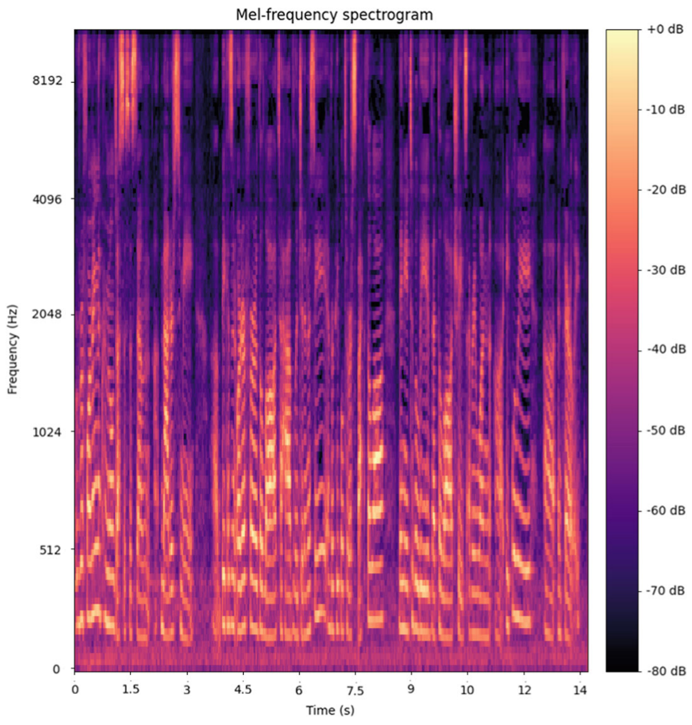

Fig. 8 Truthful image example

$$\text{F1 score} = \frac{2 * (\text{Precision} * \text{Recall})}{\text{Precision} + \text{Recall}} \tag{3}$$

5 Deception Detection Models

5.1 Conventional Models for Textual Data Only

To train our deception detection models, we explore various conventional algorithms such as:

1. Support Vector Machines (SVM) (called Model 1), and
2. K-Nearest Neighbors (KNN) (called Model 2), and
3. Logistic Regression (LG) (called Model 3).

To optimize their performance, we conduct a grid search to fine-tune the hyperparameters of each model. This thorough parameter tuning significantly improves the predictive power of our models. Table 4 lists parameters and their values obtained through the grid search. Bold values represent parameters selected by the stepwise

Table 4 Parameter lists of for grid search

Model	Parameters
SVM	C: 0.001, 0.1, **1**, 10, 100, 1000
	Kernel: **linear**, poly, rbf, sigmoid
	Gamma: **scale**, auto
LG	C: 0.001, 0.01, 0.1, 1, **10**, 100
KNN	n neighbors: 3, 5, **7**
	Weights: **uniform**, distance
	p: 1, **2**

Table 5 Explanation of parameters used in grid search

Model	Parameter explanation
SVM	**C**: Controls the trade-off between fitting the training data and generalizing to new data. Smaller values lead to a smoother decision boundary
	Kernel: Specifies the function used to transform the data (e.g., linear, polynomial, RBF, sigmoid) for better decision boundaries
	Gamma: Defines how far the influence of a single training example reaches, affecting the complexity of the model
LG	**C**: Controls the regularization strength, balancing between fitting the data closely and preventing overfitting
KNN	**n_neighbors**: Number of neighbors considered for making predictions
	Weights: Determines if all neighbors contribute equally (uniform) or if closer neighbors have more influence (distance)
	p: Defines the distance metric (Manhattan for $p = 1$, Euclidean for $p = 2$)

approach, and underlined values represent parameters selected by the OVL approach. Table 5 explains the parameters used in grid search for each model.

5.2 *Deep Models and Pre-trained Models for Textual Data Only*

5.2.1 Model 4: 1 BiLSTM

For textual models, we focus on improving deception detection using the BiLSTM model. The BiLSTM model includes three primary layers. The first layer is an embedding layer that transforms integer encoded words into dense fixed-sized vectors. The second layer is a BiLSTM layer that processes these vectors into a sequence of outputs. The output from this layer is then passed to a Dense layer with a sigmoid activation function, which outputs a single value. This value predicts the likelihood of the input text being deceptive or truthful, interpreting it as a probability between 0 and 1. We compile the model using binary cross-entropy as the loss function, utilize Adam as the optimizer, and measure performance with the accuracy metric.

5.2.2 Model 5: 1 BiLSTM + Dropout Layer

We build Model 5 on top of Model 4. We add architectural features to improve the performance of the detection of deception. A key addition is the GlobalMaxPool1D layer and the Dropout layer. The GlobalMaxPool1D layer reduces the LSTM output to a single maximum value per feature, highlighting the most important signals for classification. The model includes a Dense layer with 64 ReLU-activated neurons to analyze these reduced features. Furthermore, a Dropout layer is added with a 20% rate to prevent overfitting by randomly skipping some neuron activations during training. The model ends with a Dense output layer. It uses a sigmoid activation to give a probability estimate of deception. Like Model 4, the model still uses binary cross-entropy loss and the Adam optimizer.

5.2.3 Model 6: 1 BiLSTM + Early Stopping

Model 6 builds on the architecture used in previous models by incorporating an Early Stopping mechanism to optimize training efficiency and prevent overfitting. Early Stopping monitors the validation loss during training and stops the process if no improvement is observed for five consecutive epochs. This approach ensures that the model does not continue to learn from the training data beyond the point of beneficial returns. Therefore, the model maintains its generalizability and prevents it from learning noise and irrelevant details from the training set.

5.2.4 Model 7: BERT + Early Stopping + Dropout

Model 7 utilizes the TFBertForSequenceClassification, which is a TensorFlow 2.0 adaptation of the BERT model for sequence classification tasks. This model processes sequences of tokens, outputting a probability distribution across various labels using the ‘bert-base-uncased’ configuration. This version of BERT is pre-trained on uncased English text, enhancing its applicability to diverse text inputs. To optimize performance and handle multiclass classification, we employ Sparse Categorical Cross Entropy as the loss function and an Adam optimizer with a learning rate of 2e-5 and epsilon of 1e-08. Additionally, Early Stopping and Dropout techniques are integrated to prevent overfitting and ensure efficient training, with the model’s configuration finalized with the chosen loss function, optimizer, and performance metrics.

5.2.5 Model 8: Pretrained GPT-2 Model

Model 8 is built using a pre-trained GPT-2 architecture. We initialize it with GPT2Model.from_pretrained() to use its existing weights and increase learning efficiency. We added a linear layer (self.fc1) to the model to convert it for sequence classification. This layer takes the hidden states from GPT-2 and transforms them for classification. During the $forward()$ method process, input IDs and masks are processed by the GPT-2 model. The outputs are reshaped using $gpt\_out.view(batch\ size, -1)$ and then passed through the linear layer to get the final class predictions. Basically, we use GPT-2 as a feature extractor to transform its complex linguistic features into class predictions.

5.2.6 Model 9: Pretrained RoBERTa Model

Model 9 employs the RoBERTa architecture, configured for sequence classification using the robust PyTorch framework alongside the Hugging Face Transformers library. We utilize the 'RoBERTa-base' model along with its associated tokenizer to prepare our input corpus. Each input sequence is tokenized and then uniformly padded to maintain consistent dimensions across all data, ensuring efficient processing. These tokenized sequences, along with their respective labels, are transformed into tensors. For training, the model is optimized using an Adam optimizer with a learning rate of 2e-5. The optimization is guided by the cross-entropy loss function, which is particularly suited for classification tasks involving multiple classes.

5.3 Deep Models for Audio Data

5.3.1 Model 10: ResNet50 + Dropout

We have modified the ResNet model from its original design to suit binary classification tasks. The base ResNet50 model, with weights pre-trained on ImageNet, uses transfer learning to take advantage of features learned from visually rich datasets. We hypothesize that this approach will improve the model's ability to recognize subtle patterns in audio spectrogram data. These audio data share similarities with the image data due to their time-frequency representation. The model uses the ResNet50 base. Its top layer is removed for customization, adjusting the input shape for the task. The last 20 layers are trainable, while earlier layers keep their ImageNet weights. It includes a Global Average Pooling 2D layer, a 0.5 rate Dropout layer, and Early Stopping to prevent overfitting. A Dense layer with 1024 neurons using ReLU activation learns non-linear combinations of features. The final layer is a Dense layer with a single neuron for binary classification. With a 0.0001 learning rate, the Adam optimizer optimizes for binary cross-entropy loss, ensuring learning and generalization.

5.3.2 Model 11: VGG-16

We employ a VGG16 base pre-trained on ImageNet, excluding its top layers. All VGG16 layers are initially non-trainable, except those in the final block. Custom layers are added on top, starting with an Input layer, then passing through a GlobalAveragePooling2D layer, a Dropout layer, and a Dense layer with 1024 units and

ReLU activation. The final output layer uses a single Dense layer with sigmoid activation. The model is compiled with an Adam optimizer (learning rate of 0.00001), binary cross-entropy loss, and accuracy as a metric.

5.4 *Model 12: Late Fusion Model for Audio Data and Textual Data*

We use the ResNet50 (model 10) for audio data and the BiLSTM model (model 6) for textual data to create a late fusion model (Fig. 9). We chose these models because they provide the highest scores for their respective modalities when we experiment with different models. We change the last layers of both models so they will produce a feature vector with 128 dimensions, making sure that features from different types are represented in the same way. The vectors for each model are put together. It makes a combined feature vector with 256 dimensions. A specially created LinearW layer takes this vector and works to balance and mix the features coming from both the audio and text paths. The LinearW layer uses a group of weights that can be adjusted and which get better during training. This lets it give each set of features a certain level of importance based on what it has learned. After this fusion layer's output is ready, it goes through another thick layer with sigmoid activation to develop the ultimate prediction.

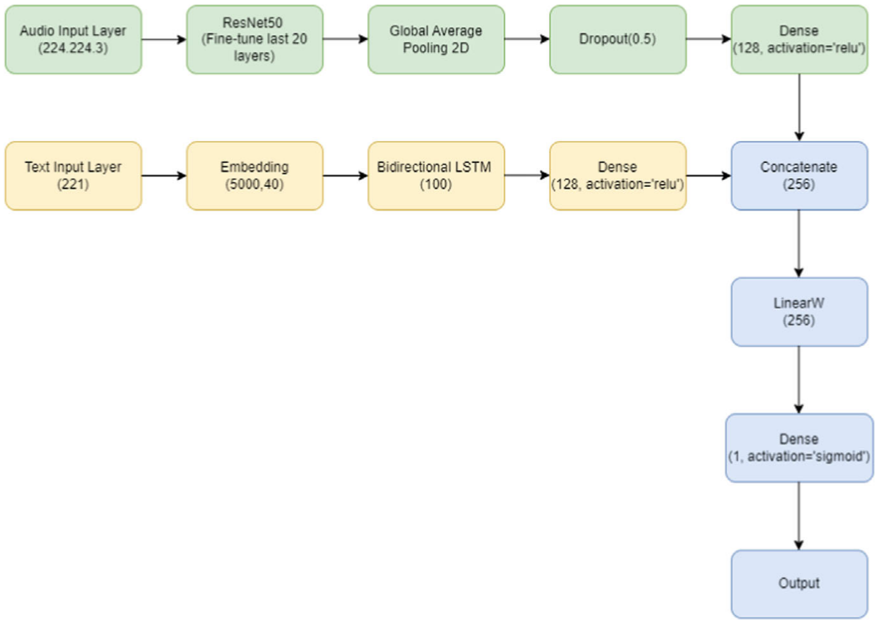
Audio Input Layer (224.224.3)
ResNet50 (Fine-tune last 20 layers)
Global Average Pooling 2D
Dropout(0.5)
Dense (128, activation='relu')
Text Input Layer (221)
Embedding (5000,40)
Bidirectional LSTM (100)
Dense (128, activation='relu')
Concatenate (256)
LinearW (256)
Dense (1, activation='sigmoid')
Output

Fig. 9 Late fusion model for textual and audio data

6 Results

6.1 *Overlapping Probability Density Functions for Linguistic Features*

PDFs are initially plotted for a selected set of features to better understand the differences in data distribution between the "Lie" and "Truth" categories. These visualizations, presented as PDFs, offer an intuitive comparison of the data distribution shapes. Smoothed PDFs are generated using Kernel Density Estimation (KDE) with a Gaussian kernel. They provide continuous representations of data distributions. These visualizations are useful for identifying features with distinct patterns that can potentially enhance the effectiveness of lie detection based on linguistic analysis.

Both PDF plots (Figs. 10, 11 and 12 in the appendix) and Table 6 visualize and report the quantitative results of the Overlapping Probability Density Functions

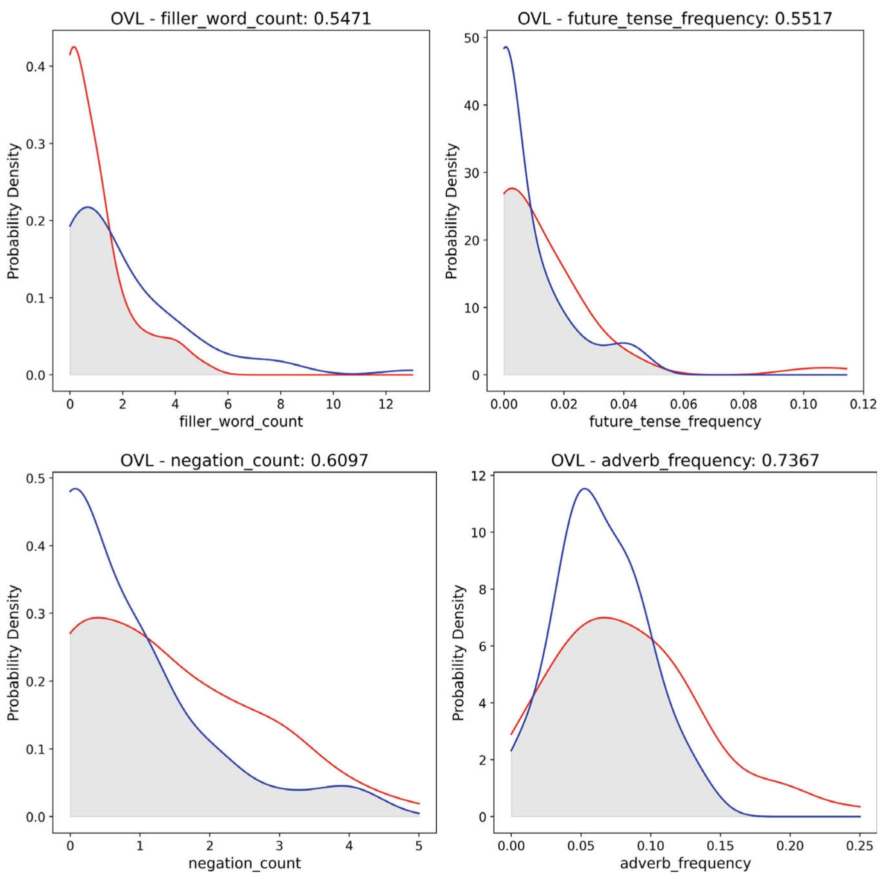

Fig. 10 PDF plots for "Lie" and "Truth" data across multiple features (part 1)

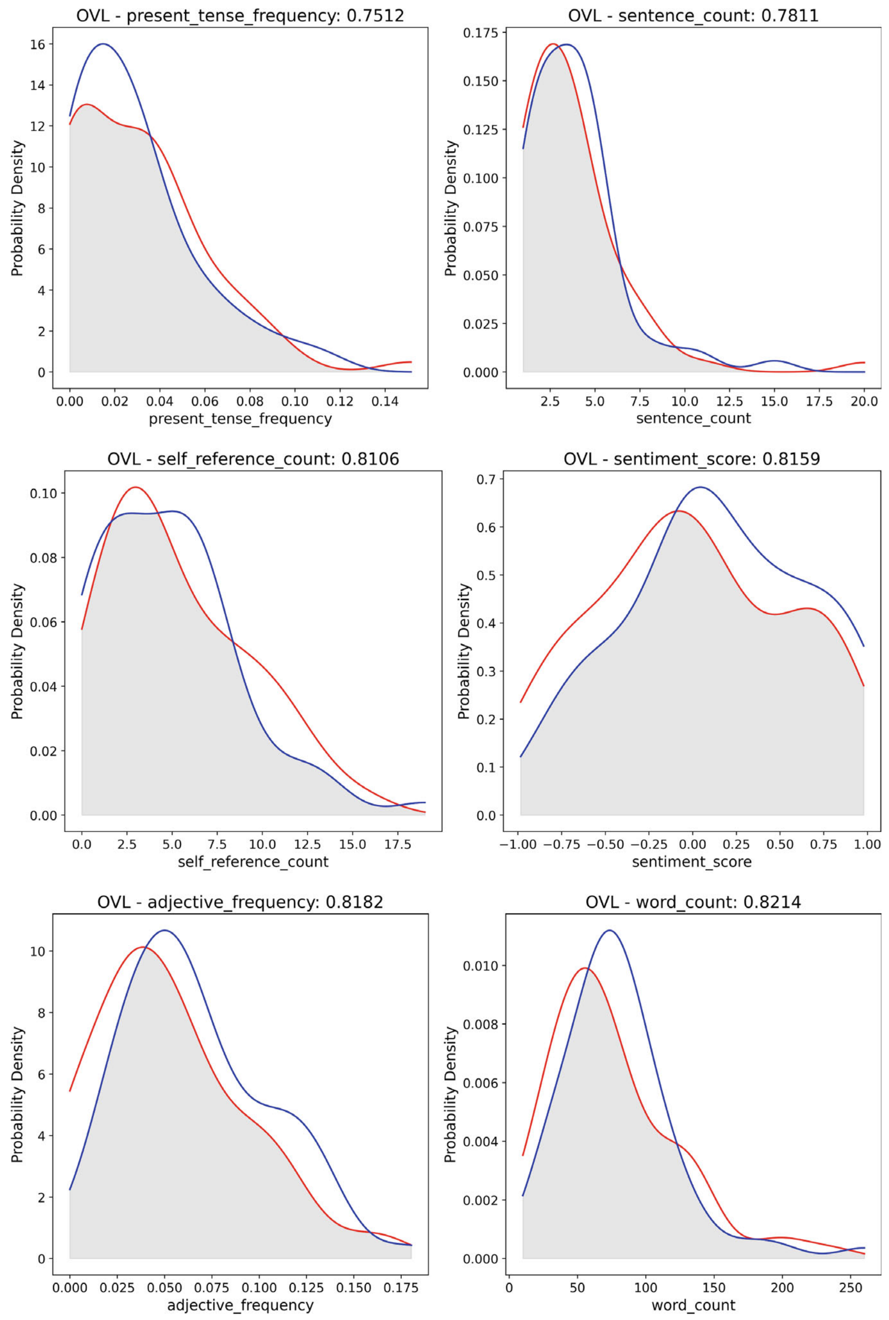

Fig. 11 PDF plots for "Lie" and "Truth" data across multiple features (part 2)

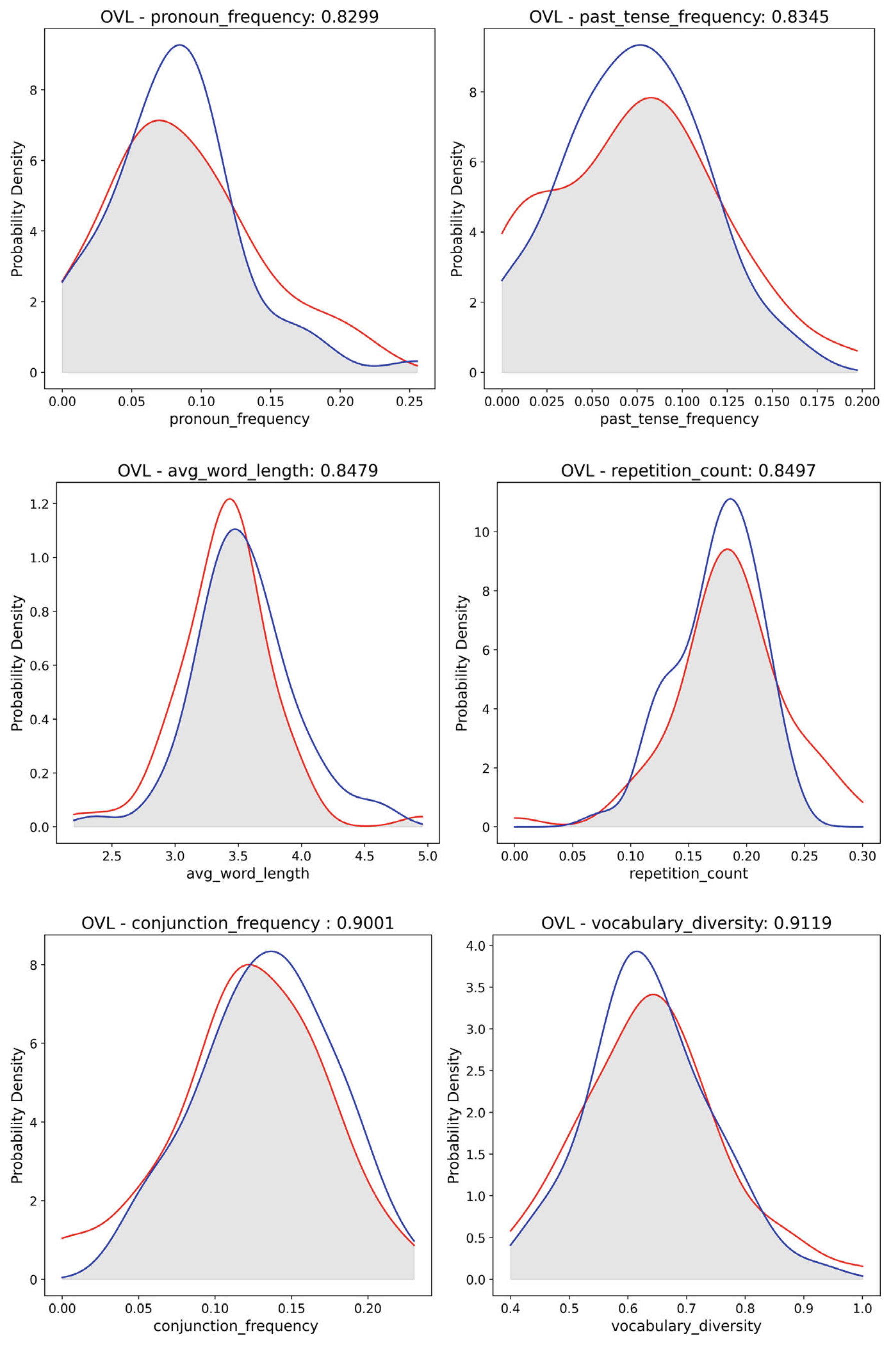

Fig. 12 PDF plots for "Lie" and "Truth" data across multiple features (part 3)

Table 6 Feature significance analysis using OVL [21]

Features	OVL score
filler_word_count	0.5471
future_tense_frequency	0.5517
negation_count	0.6097
adverb_frequency	0.7367
present_tense_frequency	0.7512
sentence_count	0.7811
self_reference_count	0.8106
sentiment_score	0.8159
adjective_frequency	0.8182
word_count	0.8214
pronoun_frequency	0.8299
past_tense_frequency	0.8345
avg_word_length	0.8479
repetition_count	0.8497
conjunction_frequency	0.9001
vocabulary_diversity	0.9119

analysis. This analysis provides a more precise measure of the discriminatory power of individual features in distinguishing between the "Lie" and "Truth" categories. By calculating the OVL scores, we can determine how much the probability density functions of different features overlap between the two categories.

As Table 6 reports, features such as "vocabulary diversity" and "conjunction frequency" exhibit high OVL scores. This indicates a substantial overlap in their probability density functions between the "Lie" and "Truth" categories. This suggests that these features may not be strong indicators on their own when it comes to distinguishing between lies and truths. On the other hand, features like "filler word count" and "negation count" display lower OVL scores, implying less overlap in their probability density functions. This indicates a higher potential for effectively distinguishing between "Lie" and "Truth" instances using these features. However, it is important to note that feature interactions and analysis context can significantly influence their discriminatory power.

6.2 *The Performance of Deception Detection Models*

6.2.1 Conventional Models for Textual Data

From the initial set of 16 features shown in Table 3, OVL and stepwise feature selection select different sets of features. For the OVL feature selection approach, we choose the threshold of 0.8, so features with an OVL score lower than 0.8 will be selected. Therefore, we have six features in total, which are (1) filter work count,

Table 7 Accuracy and F1 scores of conventional models for textual data only. All results are presented as percentages

Model	Train accuracy	Test accuracy	F1 score
Model 1a: SVM + OVL	61.12	59.37	70.44
Model 1b: SVM + Stepwise	64.46	63.77	69.8
Model 2a: KNN + OVL	70.65	63.6	65.47
Model 2b: KNN + Stepwise	71.69	62.83	63.07
Model 3a: LR + OVL	66.12	63.63	67.89
Model 3b: LR + Stepwise	**66.11**	**68.53**	**71.69**

(2) future tense frequency, (3) negation count, (4) adverb frequency, (5) present tense frequency, and (6) sentence count. In contrast, the stepwise approach carefully chose five features that showed the strongest discriminatory potential: (1) average word length, (2) vocabulary diversity, (3) frequency of adjectives, (4) frequency of adverbs, and (5) the count of filler words. These features played a crucial role in our efforts to detect deception.

Table 7 presents an evaluation of conventional models for both the OVL and the stepwise feature selection approach in terms of accuracy and F1 scores. Among the three convention models with the OVL approach, the SVM (Model 1a) has the lowest test accuracy but the highest F1 score. SVM (Model 1b) achieves a relatively lower test precision and F1 score for the stepwise approach. KNN (Model 2b) shows a reasonable training accuracy but faces challenges in generalization, with a lower test accuracy and an F1 score. Among the three different models evaluated, LR (Model 3b) stands out with its test accuracy of 68.53% and F1 score of 71.69%. These results highlight the strong potential of this model in distinguishing deceptive actions.

6.2.2 Deep Learning Models for Textual Data

Table 8 summarizes the performance of different deep models with various architectures and techniques. Model 4, with only one BiLSTM layer, shows improvements in precision (67.73%) and the F1 score (69.83%), indicating the importance

Table 8 Accuracy and F1 scores of deep models for textual data only [21]. All results are presented as percentages

Model	Train accuracy	Test accuracy	F1 score
Model 4: 1 BiLSTM	100	67.73	69.83
Model 5: 1 BiLSTM + Dropout	100	66.9	66.18
Model 6: 1 BiLSTM + Early stopping	**100**	**93.57**	**94.48**
Model 7: BERT + Early stopping + Dropout	83.54	68.73	64.63
Model 8: Pretrained GPT2 model	99.79	58.73	60.12
Model 9: Pretrained RoBERTa model	88.18	71.2	73.71

Table 9 Accuracy and F1 scores of deep models for audio data only. All results are presented as percentages

Model	Train accuracy	Test accuracy	F1 score
Model 10: ResNet50	**96.04**	**93.57**	**92.16**
Model 11: VGG16	96.01	87.63	89.25

of simplifying the model structure. Model 5 incorporates a Dropout layer alongside a single BiLSTM layer, demonstrating the impact of regularization techniques. However, its accuracy (66.9%) and F1 score (66.18%) are slightly lower than Model 5. Model 6 introduces Early Stopping and significantly enhances predictive performance. Model 6 achieves an impressive accuracy of 93.57% and an F1 score of 94.48%. This finding highlights the importance of monitoring validation loss during training to prevent overfitting. Among the three pre-trained models, Model 9 applies pre-trained Roberta, giving the highest scores. Table 8 reveals the performance variations among different models and emphasizes the importance of carefully selecting architecture and techniques. The findings further show that regularization techniques, such as Early Stopping, can help prevent overfitting and improve generalization capabilities.

6.2.3 Deep Models for Audio Data

Table 9 shows the performance of the audio models. Model 10, which employs the ResNet50 structure, gets good training and test accuracy results with 96.04% and 95.57%, respectively. It also achieves an F1 score of 92.17%. It shows it can generalize well when finding lies in audio. However, Model 11 with VGG16 structure also has good accuracy in training at 96.01% but a slight drop in test accuracy to 87.63%, and an F1 score of 89.25%. This decrease might show that even though VGG16 is very good for pulling out features, it could be worse at making these features more general than ResNet50.

6.2.4 Late Fusion Models for Both Textual and Audio Data

As mentioned earlier, we combined ResNet50 (model 10) for audio data and BiLSTM (model 6) for textual data to create the late fusion model (Fig. 9). We chose these models based on their outstanding performance in their respective modalities during our experiments. Table 10 presents how our late fusion model performed on five cross-validation folds. On average, the late fusion model obtains a test accuracy of 90.9% and an F1 score of 91.07%. The second fold shows the best results for test accuracy and F1 score, with both around 96%. On the other hand, performance is not as good in the fifth fold; it has an accuracy of about 79.12% and an F1 score near 82.76%.

Table 10 Late fusion model weights and performance across 5-fold cross validation. All results are presented as percentages

Fold	Audio weight	Text weight	Test accuracy	F1 score
1	0.49731752	0.50268245	92	90.9
2	0.5032117	0.49678826	95.83	96
3	0.516298	0.4837021	91.67	88.89
4	**0.5161787**	**0.48382124**	**95.83**	**96.77**
5	0.49938968	0.5006103	79.12	82.76
Average	0.5064791	0.4935209	90.9	91.07

7 Discussion

7.1 *Textual Models*

7.1.1 Conventional Models

The stepwise method selects a variety of linguistic features. They are the average word length, the diversity of the vocabulary used, the frequency of adjective and adverb usage, and the count of filler words. These features are chosen because they can provide valuable information on speech patterns indicative of deception. On the other hand, the OVL method focuses on a different set of features. These include the count of filler words, the frequency of adverb usage, the usage of the future tense, the frequency of negations, and the usage of the present tense. Interestingly, filler words and adverb frequency are selected by both methods. Filler words, which are often used as hesitations or distractions in speech, may be indicative of deceptive tendencies, as they could suggest that the speaker is trying to buy time or divert attention. The consistent selection of adverb frequency across both methods further suggests that the manner or intensity with which expressions are made might hold significant weight in identifying deceptive behavior. This shared focus on filler words, and adverb frequency highlights their potential importance in the study of deceptive speech patterns.

For conventional models using the OVL feature selection method, SVM (Model 1a) scores the highest F1 score at 70.44%. In contrast, when using features from the stepwise method, LG (Model 3b) achieves the best F1 score. This suggests that the performance of the models can vary according to the selected features. LG (Model 3b) attains the highest test accuracy and F1 score among all conventional models with both feature selection methods. However, LR (model 3b) also shows signs of underfitting in our analysis. The relatively low train accuracy of 66.11% implies that the model struggles to fit the training data adequately. However, the test accuracy is even higher at 68.53%. This difference between the accuracy of the train and the test is a classic indicator of inadequate fitting. This underfitting issue may be attributed to the simplicity of the LR model, which may not be able to capture complex, nonlinear relationships within the data. Consequently, the LR model's limited capacity to capture these complex patterns ultimately compromises its overall performance and prevents it from achieving higher accuracy on both the train and test sets. By exploring more sophisticated models, such as deep models, we could strive to improve our models' accuracy and generalization capabilities, ultimately enhancing our analysis's overall performance.

Şen et al. [24] also conducted a study in which they implemented conventional classifiers such as SVM and RF. They reported that their RF model achieved the highest accuracy of 64.41% using the Linguistic Inquiry and Word Count (LIWC) lexicon. When comparing it to our conventional model, our LG model (Model 3b) uses a smaller feature set and outperforms their RF model in terms of accuracy. This indicates that our feature selection approach plays a crucial role in enhancing models' performance by eliminating noise and irrelevant data.

7.1.2 Deep Models

We start building a simple model (i.e., Model 4), which consists of a single BiLSTM layer. Since its test accuracy is much lower than its train accuracy, this Model 4 shows signs of overfitting. This means that Model 4 does not perform well on new data. To improve and manage overfitting more effectively, we progressively integrate additional layers and techniques, such as Dropout and Early Stopping, into subsequent models. Dropout and Early Stopping are important techniques in deep learning for managing overfitting and improving model performance. Dropout randomly removes certain units during training to balance network weights [25]. However, when we add Dropout to Model 5, it does not perform better than Model 4. It has similar training accuracy but lower test accuracy and F1 score. This means Dropout does not successfully control overfitting or help Model 5 generalize to the test data. Model 6, which is Model 4 with added early stopping, performs much better than Models 4 and 5. It has a test accuracy of 93.57% and an F1 score of 94.48%. While it keeps the high training accuracy of Model 4, Model 6 has much better test accuracy and F1 score because of early stopping. Early Stopping adjusts the number of epochs in backpropagation and forward propagation to prevent overfitting and find the best point for model performance [25]. These results show that the right regularization techniques are key to managing overfitting and getting the best performance from a model. Model 6's success shows how useful early stopping can be in deep learning. It is a good choice for applications that need simple, high-performing models. Pretrained models are not the best option for this task. The value of a model, whether it's pretrained or not, depends on the dataset and the nature of the task.

7.2 Audio Models

After looking at the study by Sehrawat et al. [23] on finding deception with CNNs, we realize that when comparing ResNet50 and VGG16 structures, ResNet50 does a better job working with audio data. Our experimental findings match this result, showing the ResNet50 model works better than the VGG16. Another reason we chose ResNet50 instead of VGG16 is because it takes much considerably less time to train. These findings imply that the ResNet50 model outperforms the VGG16 model in terms of accuracy and computational efficiency. Therefore, we decide to use the ResNet50 architecture in our late fusion model.

7.3 Late Fusion Model

Table 1 includes a broader range of referenced works, including those using video data. Since our study focuses only on text and audio, we limit comparisons in Table 11 to models using text and audio. Table 11 shows that our late fusion model outperforms the previous model in both test accuracy and F1 scores. With 90.9% in accuracy and an F1 number of 91.1%, our model does better compared to Sehrawat et al.'s method with 80% accuracy, and Zhang et al.'s work has a correct rate of 84.40%. Given that the dataset is nearly balanced, we expect minimal differences between accuracy and F1 score, and our model's performance aligns with this expectation.

We see a consistent balance between audio and text inputs when we carefully look at how the last layer of fusion gives weights over five validation folds (as shown in Table 10). There is only a small change around an almost equal division. This shows that our model is strong because even little changes in the weights, which go from about 49.7–51.6% for audio and then similar for text, are good enough to handle the slight differences in each fold's information. The balance that is always the same makes sure one way of getting information does not take over so the model can use what was good about both audio and textual data. This careful way of deciding importance really helps make the model work very well and be trusted with different kinds of information because it mixes ways to get knowledge together in a smart way to find deception.

Table 11 Comparison of test accuracy and F1 score for models with different modalities

Modality	Previous research	Accuracy (%)	F1 score (%)
Text	Venkatesh et al. [31]	84	N/A
	Hsiao and Sun [13]	84	82.64
	Zhang et al. [33]	82.26	65.87
	Our model (BiLSTM)	**93.57**	**94.48**
Audio	Venkatesh et al. [31]	76	N/A
	Hsiao and Sun [13]	88	87.92
	Zhang et al. [33]	84.59	70.53
	Our model (ResNet50)	**93.57**	**92.16**
Text + Audio	Sehrawat et al. [23]	80	N/A
	Zhang et al. [33]	84.40	70.80
	Our model (late fusion)	**90.9**	**91.1**

8 Conclusion and Future Work

This work aims to create a multimodal model that uses both transcribed text and audio data to detect deceptive behavior more accurately. We extracted a total of 16 textual features and identified five highly significant ones using both the stepwise method and the Overlapping Coefficient (OVL) method. Through our experiments, Logistic Regression (LR) achieves the highest accuracy among conventional models with an accuracy of 68.53% and an F1 score of 71.69%. However, our deep learning model, a BiLSTM with Early Stopping, outperforms all other textual models, achieving an accuracy of 93.57% and an F1 score of 94.48%. For the audio data, the ResNet50 model performs exceptionally well, achieving an accuracy of 93.57% and an F1 score of 92.16%. Furthermore, by combining text and audio data through a late-fusion approach, we achieve an accuracy of 90.9% and an F1 score of 91.07%, outperforming previous research on similar datasets.

While this project focused on creating textual and audio models, further analysis is needed to explore the interactions between features from both modalities, such as the correlation between high pitch and increased wordiness with deception. In future work, we plan to add behavioral features from video and images to provide a more comprehensive understanding of deceptive behavior. It is important to note that the Real Life Trial dataset is relatively small, and while the results are promising, experiments with larger datasets will enhance the robustness and generalizability of the model. Expanding the dataset will help ensure that our findings are applicable across different contexts and populations.

Acknowledgements This research was supported by the U.S. National Science Foundation (Awards#: 2319802 and 2319803). Opinions, findings, and conclusions are those of the authors and do not necessarily reflect the views of the NSF.

References

1. Abri F, Gutiérrez LF, Namin AS, Jones KS, Sears DRW. Linguistic features for detecting fake reviews. In: 2020 19th IEEE international conference on machine learning and applications, ICMLA; 2020. p. 352–9.
2. Arshad M, Zhao D, Zare E, Sefton M, Triantafilis J. Proximally sensed digital data library to predict topsoil clay across multiple sugarcane fields of Australia: applicability of local and universal support vector machine. CATENA. 2021;196:104934.
3. Aslan M, Baykara M, Alakuş TB. LSTMNCP: lie detection from EEG signals with novel hybrid deep learning method. Multimed Tools Appl. 2023;1–17.
4. Bareeda EPF, Mohan BSS, Muneer KVA. Lie detection using speech processing techniques. J Phys Conf Ser. IOP Publishing. 2021;1921:012028.
5. Bidirectional LSTM in NLP. https://www.geeksforgeeks.org/bidirectional-lstm-in-nlp/
6. Chebbi S, Jebara SB. Deception detection using multimodal fusion approaches. Multimed Tools Appl. 2021;1–30.
7. Devlin J, Chang M-W, Lee K, Toutanova K. Bert: pre-training of deep bidirectional transformers for language understanding. 2018. arXiv:1810.04805

8. Fu H, Yu H, Wang X, Lu X, Zhu C. A semi-supervised speech deception detection algorithm combining acoustic statistical features and time-frequency two-dimensional features. Brain Sci. 2023;13(5):725.
9. Gallardo-Antolín A, Montero JM. Detecting deception from gaze and speech using a multimodal attention LSTM-based framework. Appl Sci. 2021;11(14):6393.
10. Gannon TA, Beech AR, Ward T. Risk assessment and the polygraph. In: The use of the polygraph in assessing, treating and supervising sex offenders: a practitioner's guide. 2009. p. 129–54.
11. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 770–8.
12. Hearst MA, Dumais ST, Osuna E, Platt J, Scholkopf B. Support vector machines. IEEE Intell Syst Appl. 1998;13(4):18–28.
13. Hsiao S-W, Sun C-Y. Attention-aware multi-modal RNN for deception detection. In: 2022 IEEE international conference on big data (big data). IEEE; 2022. p. 3593–6.
14. Hu S. Detecting concealed information in text and speech. In: Proceedings of the 57th annual meeting of the association for computational linguistics. 2019. p. 402–12.
15. Kopev D, Ali A, Koychev I, Nakov P. Detecting deception in political debates using acoustic and textual features. In: 2019 IEEE automatic speech recognition and understanding workshop (ASRU). IEEE; 2019. p. 652–9.
16. Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, Levy O, Lewis M, Zettlemoyer L, Stoyanov V. Roberta: a robustly optimized Bert pretraining approach; 2019.
17. Long short-term memory networks (LSTM)—simply explained! https://databasecamp.de/en/ml/lstms
18. Mallow MS. The admissibility of polygraph test as evidence. In: Proceedings of SOCIOINT, 2020(7th), 2020.
19. Marcolla FM, de Santiago R, Dazzi RLS. Novel lie speech classification by using voice stress. ICAART. 2020;(2):742–9.
20. Mendels G, Levitan SI, Lee K-Z, Hirschberg J. Hybrid acoustic-lexical deep learning approach for deception detection. In: Interspeech. 2017. p. 1472–6.
21. Nguyen T, Abri F, Namin AS, Jones KS. Deception and lie detection using reduced linguistic features, deep models and large language models for transcribed data. In: 2024 IEEE 48th annual computers, software, and applications conference, COMPSAC; 2024. p. 376–81.
22. Sarzynska-Wawer J, Pawlak A, Szymanowska J, Hanusz K, Wawer A. Truth or lie: exploring the language of deception. PLoS ONE. 2023;18(2):e0281179.
23. Sehrawat PK, Kumar R, Kumar N, Vishwakarma DK. Deception detection using a multimodal stacked bi-LSTM model. In: 2023 international conference on innovative data communication technologies and application (ICIDCA). IEEE; p. 318–26.
24. Şen MU, Perez-Rosas V, Yanikoglu B, Abouelenien M, Burzo M, Mihalcea R. Multimodal deception detection using real-life trial data. IEEE Trans Affect Comput. 2020;13(1):306–19.
25. Sitaula C, Ghimire N. An analysis of early stopping and dropout regularization in deep learning. Int J Concept Comput Inf Technol. 2017;5(1):17–20.
26. Solaiman I, Brundage M, Clark J, Askell A, Herbert-Voss A, Wu J, Radford A, Krueger G, Kim JW, Kreps S, McCain M, Newhouse A, Blazakis J, McGuffie K, Wang J. Release strategies and the social impacts of language models. 2019.
27. Talaat FM. Explainable enhanced recurrent neural network for lie detection using voice stress analysis. Multimed Tools Appl. 2023;1–23.
28. Tao H, Lei P, Wang M, Wang J, Fu H. Speech deception detection algorithm based on SVM and acoustic features. In: 2019 IEEE 7th international conference on computer science and network technology (ICCSNT). IEEE; 2019. p. 31–3.
29. Theckedath D, Sedamkar RR. Detecting affect states using VGG16, ResNet50 and SE-ResNet50 networks. SN Comput Sci. 2020;1(2):79.
30. Van Houdt G, Mosquera C, Nápoles G. A review on the long short-term memory model. Artif Intell Rev. 2020;53(8):5929–55.
31. Venkatesh S, Ramachandra R, Bours P. Robust algorithm for multimodal deception detection. In: 2019 IEEE conference on multimedia information processing and retrieval (MIPR). IEEE; 2019. p. 534–7.

32. Zhang F, Hu C, Yin Q, Li W, Li H-C, Hong W. Multi-aspect-aware bidirectional LSTM networks for synthetic aperture radar target recognition. IEEE Access. 2017;5:26880–91.
33. Zhang H, Ding Y, Cao L, Wang X, Feng L. Fine-grained question-level deception detection via graph-based learning and cross-modal fusion. IEEE Trans Inf Forensics Secur. 2022;17:2452–67.