Ain Shams University
Faculty of Computer & Information Sciences
Computer Science Department

Multimodal Deception Detection System

By

Doaa Yehia	Computer Science
Medhat Essam	Computer Science
Mena Ashraf	Computer Science
Mohamed Moawad	Computer Science
Reem Ayman	Computer Science
Rodayna Mohamed	Computer Science

Under Supervision of

Dr. Salsabil Amin,
Basic Science Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

T.A. Zeina Rayan,
Computer Science Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

July 2024

Acknowledgement

We would like to take this opportunity to express our gratitude to everyone who supported us throughout this project. We deeply appreciate their essential guidance, constructive feedback, friendly advice, and their genuine and insightful perspectives.

We are particularly grateful to our supervisor, Dr. Salsabil Amin, for her invaluable help and support. We also extend our sincere thanks to T.A. Zeina Rayan for guiding us throughout the project; her assistance was invaluable.

Abstract

Deception is prevalent in our daily lives, often with serious consequences in critical areas such as police investigations, airport security, and court-room trials. Human ability to detect deception without special tools is very limited.

Deceptive behaviour often triggers multiple cues in individuals, encompassing both verbal and visual signals. For instance, deceptive individuals may exhibit inconsistencies in their linguistic choices, changes in their tone of voice, and hesitation. Visually, they might display nervous gestures, avoid eye contact, or exhibit facial expressions that are incongruent with their words. Detecting these cues is complex and typically requires professional training, which can be costly and time-consuming, and even then, accuracy is not guaranteed.

Our goal was to develop an AI based system that can leverage visual and verbal cues to identify deception. This solution aims to be more cost-effective than hiring professionals and beneficial in various fields, including police investigations.

In our research, we utilized a multimodal approach, incorporating video data, audio signals, linguistic features from text, and micro-expressions to enhance lie detection. We explored previous benchmarks and tried to enhance them. We worked from scratch to build similar models and test ourselves to see the effect of multimodal features combined. Our system achieved an accuracy of **95.8%** on the Real-life Deception Detection Dataset using late fusion on visual, vocal and linguistic features.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Stands for
ADAM	Adaptive Moment Optimization
ANN	Artificial Neural Network
AU	Action Units
Bi-LSTM	Bidirectional Long Short-Term Memory
CNN	Convolutional Neural Network
Conv	Convolutional
Convnet	Convolutional network
CPU	Central Processing Unit
EST	Emotional State-based features
FACS	Facial Action Coding System
FC	Fully-Connected
GloVe	Global Vectors for Word Representation
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
IS13	Interspeech 2013 ComParE features
LSTM	Long Short-Term Memory
ME	Multimodal Embeddings
MFCC	Mel Frequency Cepstral Coefficients
ML	Machine Learning

MLP	Multi-Layer Perceptron
MTCNN	Multi-Task Cascaded Convolutional Networks
NLP	Natural Language Processing
NN	Neural Network
OS	Operating System
RAM	Random Access Memory
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RGB	Red Green Blue
RMSprop	Root Mean Square Propagation
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
TF-IDF	Term Frequency-Inverse Document Frequency
UI	User interface
XGBoost	Extreme Gradient Boosting

1. Introduction

Deception involves actions or statements intended to mislead, conceal the truth, or promote false beliefs or ideas. While everyday lies often have minor consequences, the ability to detect deception is critical in contexts where security is paramount, such as police investigations and airport security.

1.1 Problem Definition

The ramifications of wrongly accusing innocent individuals or allowing the guilty to go free are severe. Thus, there is a pressing need for a reliable and efficient system to detect deceptive behaviour and distinguish between liars and truth-tellers. Traditional methods, such as the polygraph test, have long been used in criminal investigations to identify deceptive behaviour [7]. However, these methods often require skin-contact devices and human expertise, making them impractical in certain situations. Additionally, the outcomes can be influenced by human error and bias [6]. Offenders can also employ countermeasures to deceive these devices and experts.

Given the limitations of traditional polygraph-like methods, learning-based approaches have been proposed to address deception detection through various modalities, including text [13] and speech [8]. Research has shown that facial expressions play a significant role in identifying deception. Micro-expressions, which are brief involuntary facial expressions, can be indicative of deceptive behaviour [2].

Modern deception detection systems have evolved to incorporate multiple modalities, leveraging advancements in technology. The effectiveness of using multimodal approaches that integrate text, audio, and visual data to improve the accuracy of deception detection has been demonstrated in various studies [5], [10], [11]. Additionally, incorporating facial affect representations in automated multimodal deception detection systems is also demonstrated in previous studies. [9]

Considering these developments, our research focuses on developing a comprehensive multimodal deception detection system. By utilizing video, audio, linguistic features from text, and micro-expressions, we aim to create a robust solution capable of accurately identifying deceptive behaviour. This system not only addresses the shortcomings of traditional methods but also offers a more holistic approach to deception detection in various critical settings, such as police investigations and airport security.

1.2 Motivation

Currently, fields such as law enforcement and airport security rely heavily on human resources for conducting investigations. Studies indicate that the ability of humans to detect deception without special aids is only about 54%, which is quite low [1]. This highlights the need for a more accurate Deception Detection system.

Detecting deception is inherently challenging for humans. It can take years of training to develop the skills necessary to identify deceit, and even then, success is not guaranteed. Various factors, such as personal biases or external influences like bribery, can affect human judgment.

With the aid of artificial intelligence, we can significantly enhance deception detection. By employing well-trained models, we can achieve accuracy levels that surpass human performance. These models can analyse facial micro-expressions from pre-recorded videos, providing a more reliable and unbiased assessment of deceptive behaviour.

1.3 Objectives

- Build a fully automated user-friendly system to detect deception.
- Train machine learning and deep learning models on different data modalities.
- Utilize multimodal data fusion for integrating different data modalities.
- Minimize false accusations of the innocent.

- Develop a reliable non-invasive alternative to traditional polygraph machines.

1.3.1 What was done to achieve these objectives

- Built a user-friendly website to upload videos and get predictions.
- Trained various neural networks to detect deception from visual and verbal cues.
    - 3D CNN
    - CNN-LSTM
    - Bi-LSTM
    - ANN
- Trained Machine learning models on the extracted micro-expressions and text.
    - SVM
    - Decision Tree
    - Random Forest
    - XGBoost

1.4 Time Plan

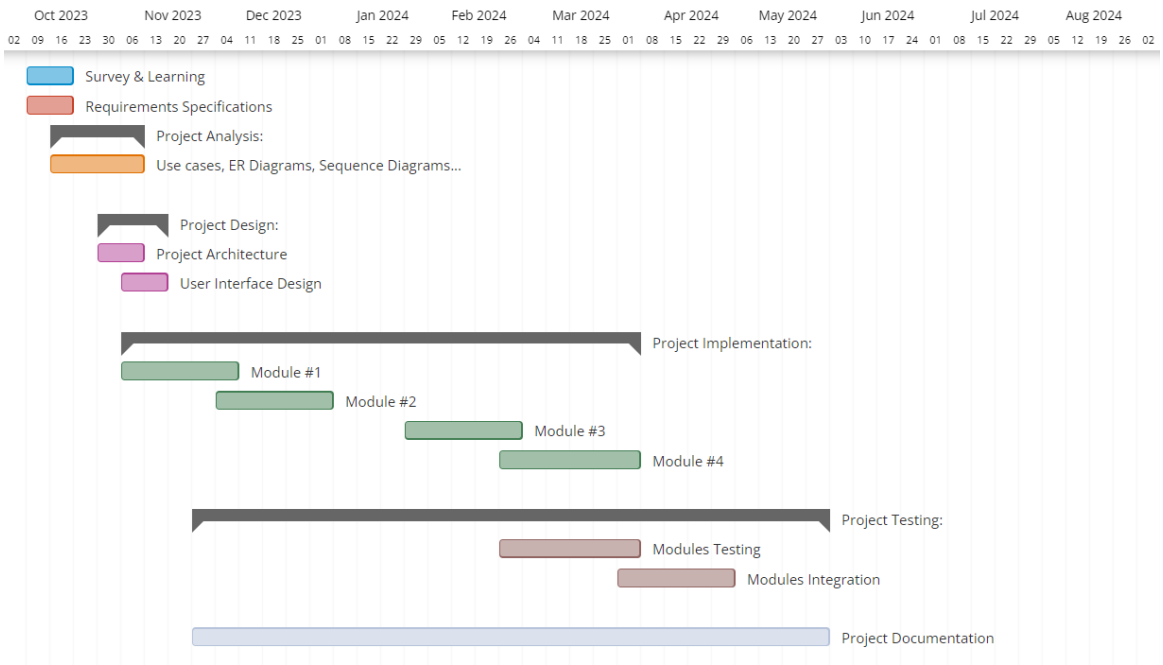

Figure 1.1: Time Plan

1. 2 weeks for survey and research.
2. 2 weeks for requirement specification.
3. 1 month for performing project analysis.
4. 2 weeks for designing project architecture.
5. 2 weeks for user interface design.
6. 4 months for modules implementation.
7. 2 months for modules integration and testing.
8. Working on documentation in parallel throughout whole project development.

1.5 Document Organization

Chapter 2: Background

This chapter provides background information about the project, including basic concepts and related research work.

Chapter 3: Analysis & Design

This chapter gives an overview of the entire system, including the target users, system architecture, analysis, and design.

Chapter 4: Dataset
This chapter provides an overview of the dataset used in the system.

Chapter 5: Implementation
This chapter details the system functions and describes the implementation of the project's modules.

Chapter 6: Results
This chapter presents the experiments conducted to improve the system's results.

Chapter 7: User Manual
This chapter explains the necessary packages and libraries that must be installed before using the application and guides the user on how to use it.

Chapter 8: Conclusion & Future Work
This chapter summarizes the conclusions and results of our work and discusses potential future work based on this project.

References
This chapter lists all the papers and sources we used and studied throughout our work.

2. Background

2.1 Neural Networks

Neural networks are a class of algorithms inspired by the human brain, designed to recognize patterns in data. These networks process sensory data through a form of machine perception, enabling the labelling or clustering of raw inputs. The patterns identified by neural networks are numerical, existing within vectors. Consequently, all real-world data, whether images, sounds, texts, or time series, must be translated into these vectors.

Neural networks and deep learning are significant topics in Computer Science and the technology industry. They currently offer the most effective solutions for numerous challenges, including image recognition, speech recognition, and natural language processing [24].

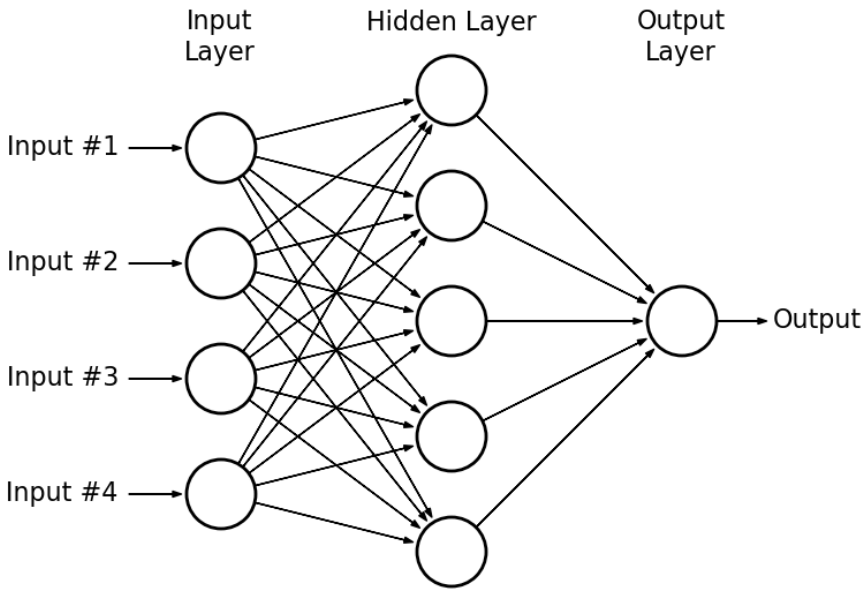

Figure 2.1: Neural Network General Architecture

The basic unit of the human brain is the neuron, while the fundamental building block of an artificial neural network is the perceptron. Perceptrons perform simple signal processing tasks and are interconnected to form a large mesh network.

A computer with a neural network is trained to perform tasks by analyzing pre-labeled training examples. For instance, in an object recognition task, a neural network is presented with numerous images of specific objects, such as cats or street signs. By identifying recurring patterns in these images, the neural network learns to categorize new images accurately.

Unlike other algorithms, neural networks with deep learning cannot be explicitly programmed for specific tasks. Instead, they must learn the information, similar to how a child's brain develops. Learning strategies for neural networks include three primary methods:

- **Supervised Learning:** In this straightforward strategy, the neural network is trained using a labeled dataset. The algorithm iteratively adjusts until it can accurately process the dataset to achieve the desired result.
- **Unsupervised Learning:** This method is used when a labeled dataset is unavailable. The neural network analyzes the data, and a cost function indicates how far the results deviate from the target. The network then adjusts to improve the algorithm's accuracy.
- **Reinforcement Learning:** In this approach, the neural network is rewarded for positive results and penalized for negative outcomes, encouraging it to learn and improve over time [24].

2.1.1 Neural Network Architecture

The fundamental unit of computation in a neural network is the neuron, also known as a node or unit. It receives inputs from other nodes or from an external source and computes an output. Each input is assigned a weight (w) based on its relative importance compared to other inputs. The node then applies a function to the weighted sum of its inputs.

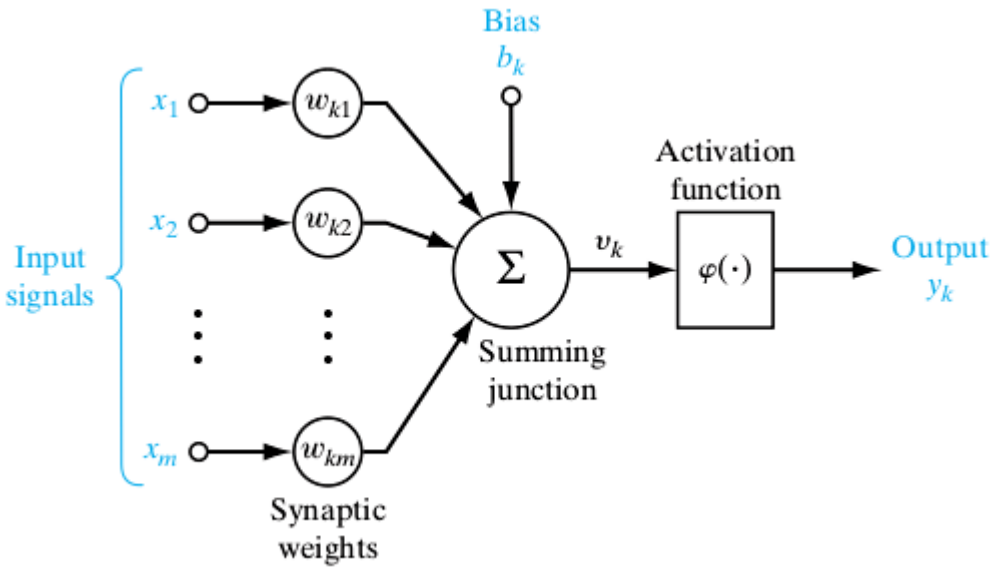

Figure 2.2: Single Neuron Architecture

Input Nodes (Input Layer): No computation occurs in this layer; it simply passes information to the next layer, which is typically the hidden layer. A group of nodes is referred to as a layer.

Connections and Weights: The network is made up of connections, each transferring the output of one neuron (i) to the input of another neuron (k). In this context, neuron i is the predecessor of neuron k, and neuron k is the successor of neuron i. Each connection is assigned a weight W_{ki}.

Output Nodes (Output Layer): This layer applies an activation function to produce the desired output format, such as the softmax function for classification tasks.

Activation Function: This function determines the output a node will generate based on its input. The activation function is defined at the layer level and applies to all neurons in that layer. It can be linear or nonlinear. Nonlinear activation functions enable neural networks to solve complex problems with only a small number of nodes. The activation function is also referred to as the transfer function.

Hidden Nodes (Hidden Layer): Hidden layers are where intermediate processing or computation occurs. They perform computations and transfer the weights (signals or information) from the input layer to the next layer, which can be another hidden layer or the output layer. A neural network can exist without a hidden layer, though hidden layers are common in more complex networks.

Learning Rule: The learning rule is an algorithm that adjusts the parameters of the neural network so that a given input produces the desired output. This process typically involves modifying the weights and thresholds within the network.

2.1.2 Types of Artificial Neural Networks

2.1.2.1 Feed Forward Neural Networks

Feedforward neural networks are artificial neural networks where the connections between units do not form a cycle. They were the first type of artificial neural network invented and are simpler than their counterpart, recurrent neural networks. They are called *feedforward* because information only travels forward in the network, and there is no feedback (loops); *i.e.*, the output of any layer does not affect that same layer. The propagation traverses first through the input nodes, then through the hidden nodes (if present), and finally through the output nodes.

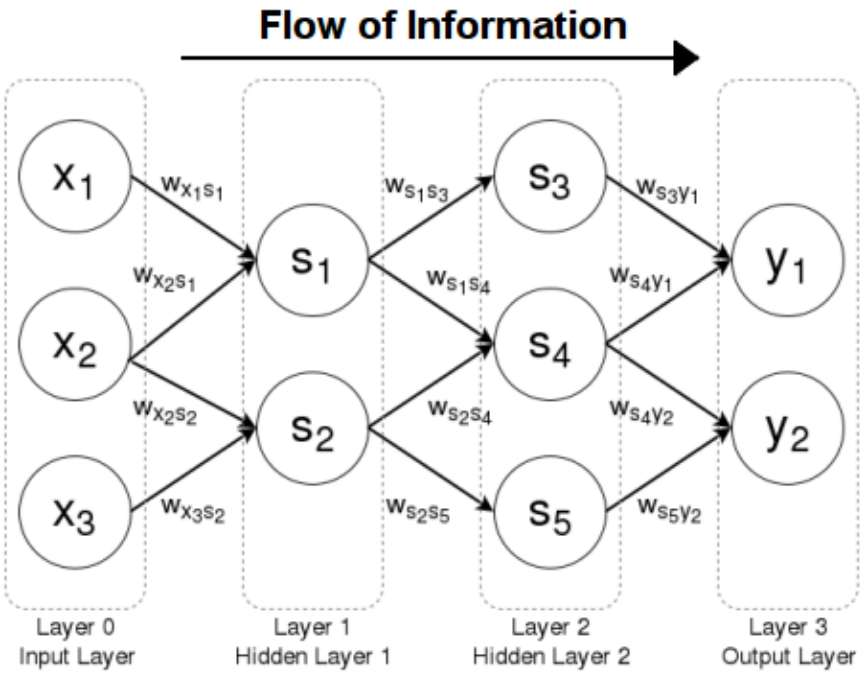

Figure 2.3: Feed Forward Neural Network

Feedforward neural networks are primarily used for supervised learning in cases where the data to be learned is neither sequential nor time-dependent. They are extensively used in pattern recognition.

2.1.2.1.1 Single-Layer Perceptron

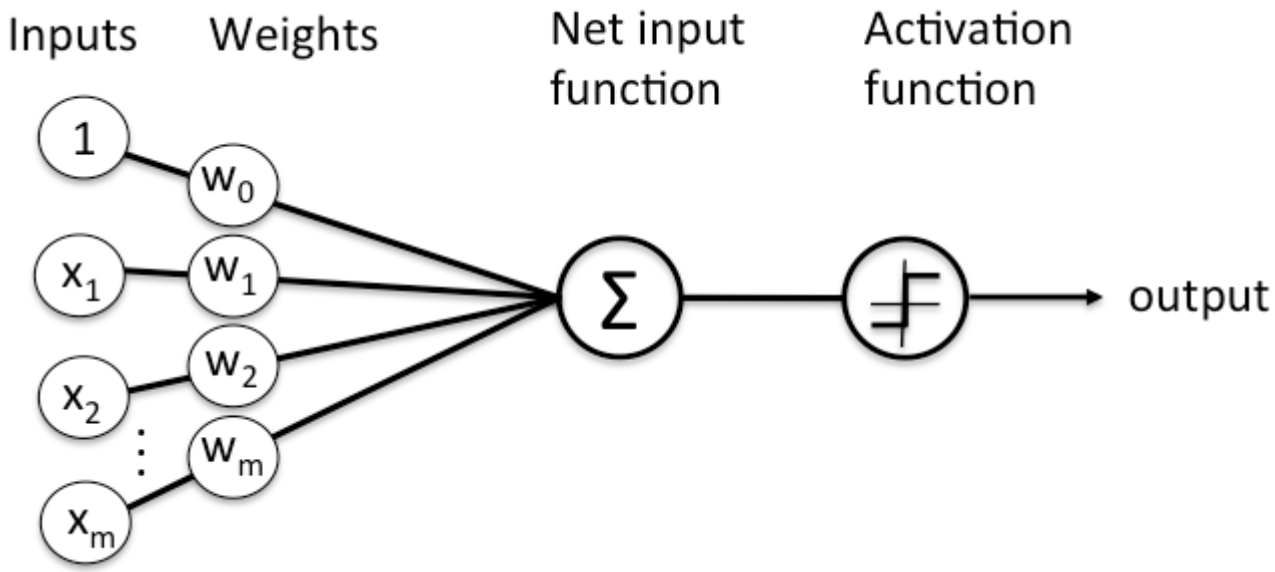

Figure 2.4: Single Perceptron

The simplest type of feedforward neural network is the perceptron, a feed-forward neural network with no hidden units. Thus, a perceptron has only an input layer and an output layer. The output units are computed directly from the sum of the product of their weights with the corresponding input units, plus some bias.

Historically, the perceptron's output has been binary, meaning it outputs a value of 0 or 1. This is achieved by passing the aforementioned product sum into the step function φ(x). This is defined as:

$$\phi(x) = \begin{cases} 1 & ,if\ x \geq 0 \\ 0 & ,if\ x\ < 0 \end{cases}$$

Since the perceptron divides the input space into two classes, 0 and 1 depending on the values of weight vector and bias, it is known as a Linear classifier. The line separating the two classes is known as the classification boundary. In the case of two-dimensional input (as in figure 2.5 below) it is a line, while in higher dimensions this boundary is a hyperplane. The weight vector defines the slope of the classification boundary while the bias defines the intercept.

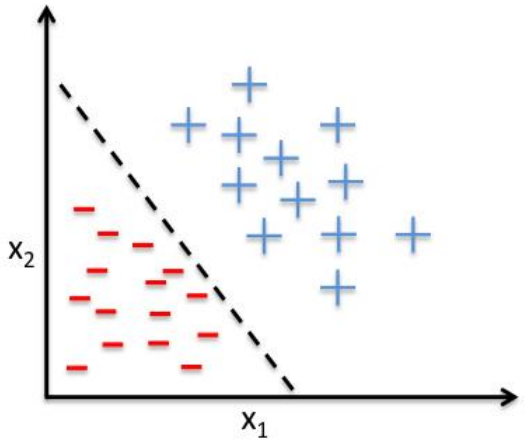

Figure 2.5: Linear Classifier

Single-layer perceptrons are linear classifiers, meaning they can only learn linearly separable patterns. However, since many functions are not linearly separable, an alternative called multi-layer perceptrons was developed.

2.1.2.1.2 Multi-Layer Perceptron (MLP)

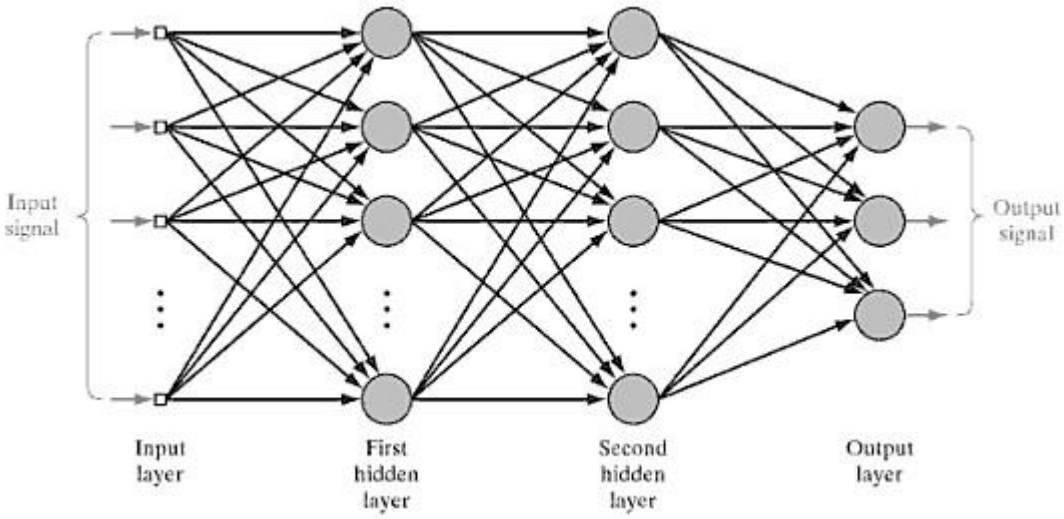

Figure 2.6: Mult-Layer Perceptron

It is an artificial neural network composed of many perceptrons. Unlike single-layer perceptrons, MLPs are capable of learning to compute non-

linearly separable functions. Because they can learn non-linear functions, they are one of the primary machine learning techniques for both regression and classification in supervised learning (in most cases the data presented to us is not linearly separable).

These neural networks consist of multiple layers of computational units, usually interconnected in a feed-forward way. Each neuron in one layer has directed connections to the neurons of the subsequent layer. In many applications, the units of these networks apply a sigmoid function as an activation function.

2.1.2.2 Recurrent Neural Networks

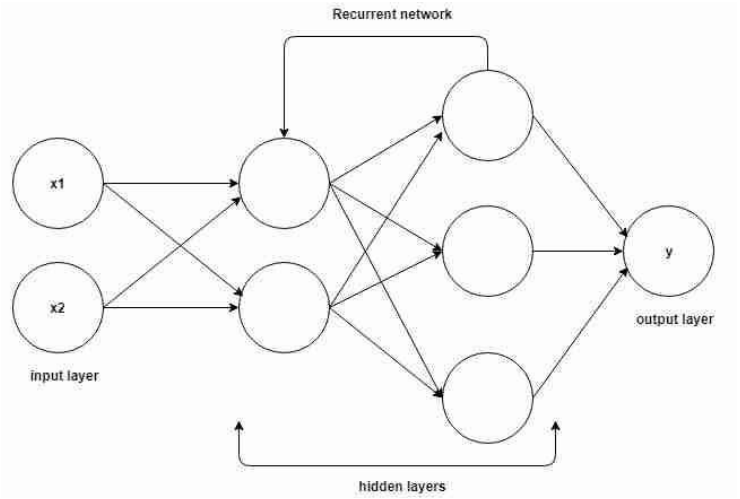

Figure 2.7: Recurrent Neural Network

Recurrent Neural Networks (RNNs) add an interesting twist to basic neural networks. A Recurrent Neural Network remembers the past and its decisions are influenced by what it has learned from the past. Note: Basic feedforward networks “remember” things too, but they remember things they learned during training. For example, an image classifier learns what a “1” looks like during training and then uses that knowledge to classify things in production. While RNNs learn similarly while training, in addition, they remember things learned from prior input(s) while generating output(s).

2.1.3 Backpropagation

The essence of neural network training lies in fine-tuning the weights based on the error rate (i.e., loss) obtained in the previous epoch (i.e., iteration). Proper tuning of the weights ensures lower error rates, enhancing the model's reliability by improving its generalization. Backpropagation involves three main steps:

- **Forward-propagation**
  1- Getting the weighted sum of inputs of a particular unit.
  2- Plugging the value we get from step 1 into the activation function and using the activation value we get (i.e. the output of the activation function) as the input feature for the connected nodes in the next layer.

$$net_j(t) = \sum_{i=0}^{n} w_{ji}(t)\, x_i(t) \equiv w_j^T . X \qquad (2.1)$$

n: the number of output units
w(j)(i): the weight of the connection from neuron i to neuron j
x(i): neuron i of the input layer

$$net_k(t) = \sum_{j=0}^{d} w_{kj}(t)\, z_i(t) \equiv w_k^T . Z \qquad (2.2)$$

w(k)(j): the weight of the connection from neuron j to neuron k
z(j): neuron j of the hidden layer

- **Backpropagation**
  We compute ‘error signal’, propagating the error backwards through the network starting at output units (where the error is the difference between actual and desired output values).

Hidden-to-Output units:

$$\Delta w_{kj}(t) = -\eta \frac{\partial E}{\partial w_{kj}} = \eta(d_k - y_k) f'(net_k) z_j$$

η: learning rate
d(k): desired output
y(k): actual output
f'(net): derivative of the activation function
z(j): neuron j of the hidden layer

Input/Hidden-to-Hidden units:

$$\Delta w_{ji}(t) = \eta \delta_j(t) x_i(t) = \eta f'(net_j) \left[\sum_{k=0}^{m} \delta_k w_{kj} \right] x_i$$

η: learning rate
δ: local gradient of neuron
f'(net): derivative of the activation function
x(i): neuron i of the input layer

- **Update weights**
  1- The weights are updated using the following rules

$$W_{new} = W_{old} - \alpha \underbrace{\frac{dJ}{dW}}_{\text{gradient}}$$

2.2 Deep Neural Networks

A deep neural network is an artificial neural network with multiple hidden layers between the input and output layers. In this architecture, the output of each hidden layer serves as the input for the next hidden layer, except for the last hidden layer, whose output flows to the output layer [24].

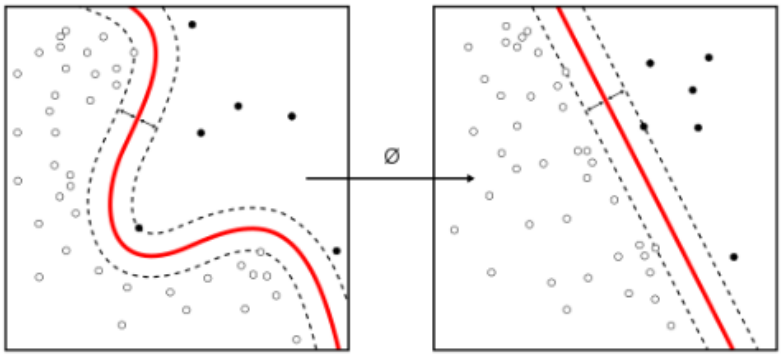

Figure 2.8: Non-linearly separable data to linearly separable data

This architecture makes data more separable, meaning that non-linearly separable data can become linearly separable through the application of certain nonlinear functions. The architecture of deep neural networks is also beneficial for extracting features from images and data points, such as in object and face recognition tasks. It adapts its weights based on the effectiveness of each pixel (or value) in achieving the desired output.

2.3 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are similar to ordinary neural networks, consisting of neurons with learnable weights and biases. The entire network still represents a single differentiable score function, mapping raw image pixels at one end to class scores at the other. Additionally, they have a loss function (e.g., SVM/Softmax) on the last (fully-connected) layer, and all the techniques developed for learning regular neural networks still apply.

The key difference is that ConvNet architectures explicitly assume that the inputs are images, allowing us to encode certain properties into the architecture. This assumption makes the forward function more efficient to implement and significantly reduces the number of parameters in the network

CNNs take advantage of the fact that the input consists of images by structuring the architecture in a more logical manner. Unlike regular neural networks, the layers of a ConvNet have neurons arranged in three dimensions: **width**, **height**, and **depth**. Note that "depth" here refers to the third dimension of an activation volume, not the total number of layers in the network.

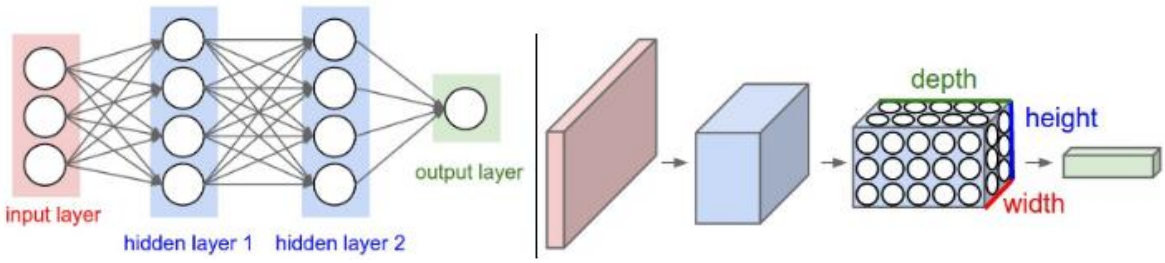

Figure 2.9: Comparison between MLP and CNN

There are three main types of layers to build ConvNet architectures: **Convolutional Layer**, **Pooling Layer**, and **Fully-Connected Layer**. These layers stacked form a ConvNet **architecture**.

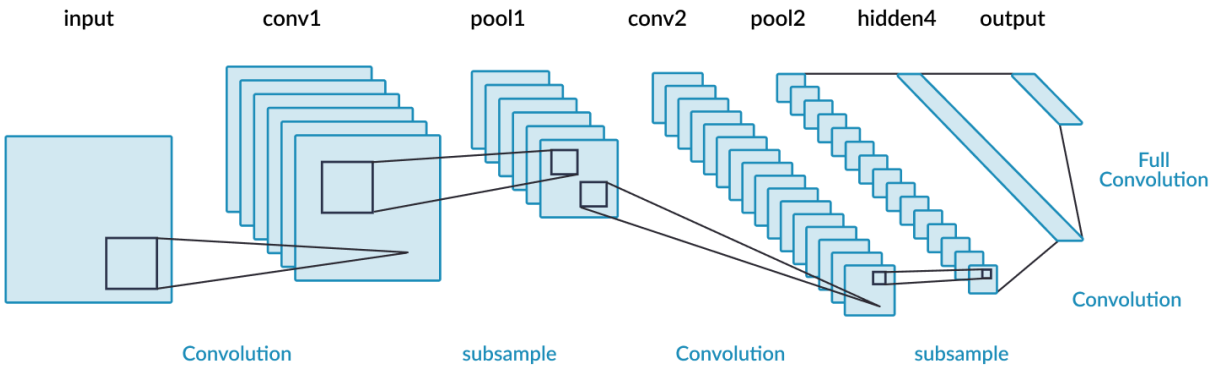

Figure 2.10: CNN Architecture

2.3.1 Convolution Layer

The Convolutional (Conv) layer is the core building block of a Convolutional Neural Network (CNN) that performs the bulk of the computational work. The first layer in a CNN is always a **Convolutional layer**. The input to this layer is a 3-dimensional volume comprising width, height, and depth.

The Conv layer contains a **filter** (or kernel), which is an array of numbers (weights) that covers a small area of the input volume, known as the receptive field. As the filter **convolves** over the input volume, it multiplies the filter values by the corresponding values in the receptive field. This process generates a single number for each unique location on the input volume.

Each convolution with a filter results in an **activation map** or **feature map**. The depth of the filter must match the depth of the input volume. By using multiple filters, we can produce multiple feature maps, increasing the depth of the output volume.

The purpose of Conv layers is to detect features in images, ranging from low-level features (such as straight edges, simple colors, and curves) to higher-level features (such as hands, paws, or ears) as the network goes deeper.

There are two main parameters that can be adjusted to modify the behavior of each Conv layer: the stride and the padding values. After selecting the filter size, these parameters determine how the filter moves across the input and how the edges are handled.

There are 2 main parameters that we can change to modify the behaviour of each Conv layer. After we choose the filter size, we also have to choose the **stride** and the **padding** values**.**

2.3.2 Strides

Controls how the filter convolves around the input volume. The amount by which the filter shifts is the stride (shifts in the width and height dimensions).

2.3.3 Padding

Padding is used to ensure that the width and height of the output volume remain the same as the input volume by padding the input volume with zeros. As Conv layers are applied, the size of the volume tends to decrease more rapidly than desired. Padding addresses this issue by preserving the spatial dimensions of the volume.

Additionally, padding helps to retain as much information about the original input volume as possible in the early layers of the network, enabling the extraction of low-level features.

2.3.4 Pooling Layer

In ConvNet architectures, it is common to periodically insert a Pooling layer between successive Conv layers. The function of the Pooling layer is to progressively reduce the spatial size of the representation, thereby decreasing the number of parameters and computations in the network and helping to control overfitting.

Spatial Pooling can be of different types, including Max Pooling, Average Pooling, and Sum Pooling.

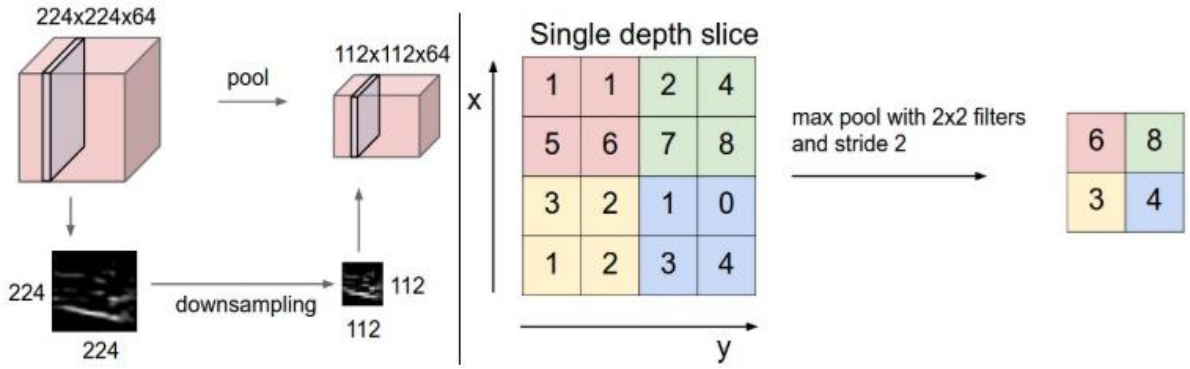

Figure 2.11: Max Pooling Layer

2.3.5 Activation Functions

The activation function is applied at the end of or between network layers to help determine if a neuron should fire. It adds a non-linear transformation to the output signal of the layer, and this transformed output is then sent as input to the next layer of neurons.

2.3.5.1 Hyperbolic Tangent function (Tanh)

The Tanh function, an improved version of the logistic sigmoid, has a range from -1 to 1 and is commonly used in feedforward networks. Its advantage is that negative inputs are mapped to strong negative values and zero inputs are mapped near zero in the Tanh graph.

The Tanh function is differentiable and monotonic, although its derivative is not monotonic. It is primarily used for binary classification tasks.

Figure 2.12: Hyperbolic Tangent Function

2.3.5.2 Non-Linearity (ReLU)

The Rectified Linear Unit (ReLU) function is the most widely used activation function in neural networks today. One of its significant advantages over other activation functions is that it does not activate all neurons simultaneously.

As shown in the image of the ReLU function below, it converts all negative inputs to zero, preventing the neuron from activating. This selective activation makes ReLU computationally efficient, as fewer neurons are activated at any given time. Additionally, ReLU does not saturate in the positive region. In practice, ReLU converges six times faster than the Tanh and sigmoid activation functions.

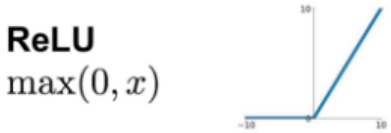

Figure 2.13: Rectified Linear Unit

2.3.6 Fully Connected Layer

In the fully connected (FC) layer, the output matrix from the previous layers is flattened into a one-dimensional vector. This vector is then fed into the fully connected layer, where each neuron is connected to every neuron in the preceding layer, similar to a traditional neural network. This layer helps in combining features learned by previous layers to make final predictions.

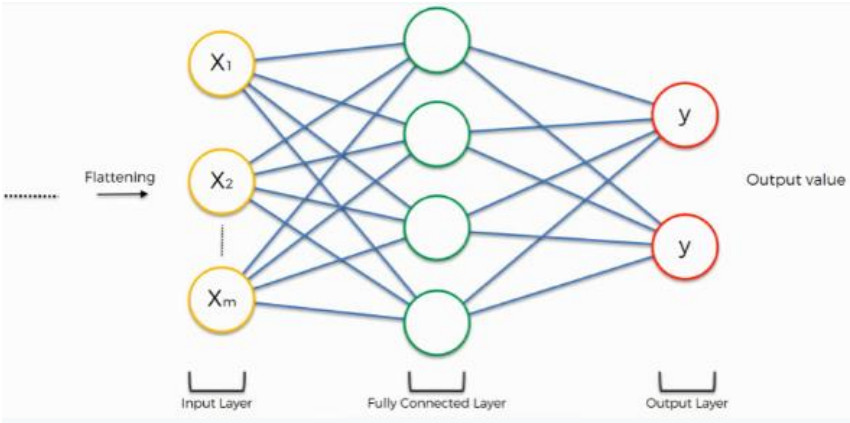

Figure 2.14: Fully Connected Layer

2.3.7 Dropout Layer

Dropout is a technique used to prevent a model from overfitting. It works by randomly setting the outgoing edges of hidden units to zero at each update during the training phase. The term "dropout" refers to the process of temporarily dropping out units in a neural network.

This technique prevents overfitting and provides a way to efficiently approximate the combination of exponentially many different neural network architectures. When a unit is dropped out, it is temporarily removed from

the network, along with all its incoming and outgoing connections. The units to be dropped are chosen randomly based on a predetermined drop-out probability.

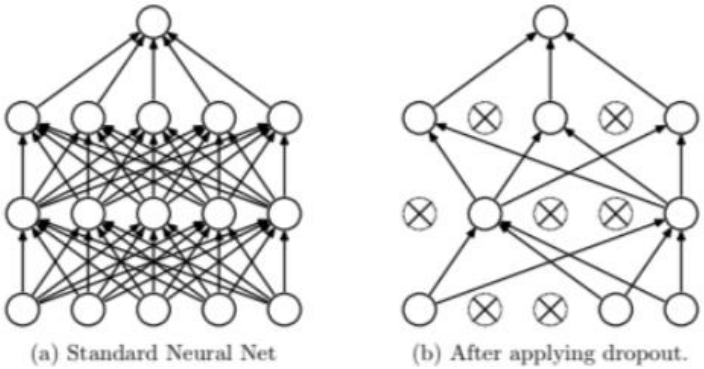

Figure 2.15: Dropout Layer

2.3.8 3D Convolutions

3D convolutions apply a three-dimensional filter to the dataset, moving in the x, y, and z directions to calculate low-level feature representations. The output shape is a three-dimensional volume, such as a cube or a cuboid. These convolutions are particularly useful for tasks such as event detection in videos and analysing 3D image data, like Magnetic Resonance Imaging (MRI) scans. Although primarily designed for 3D data, 3D convolutions can also be applied to 2D data inputs, such as images.

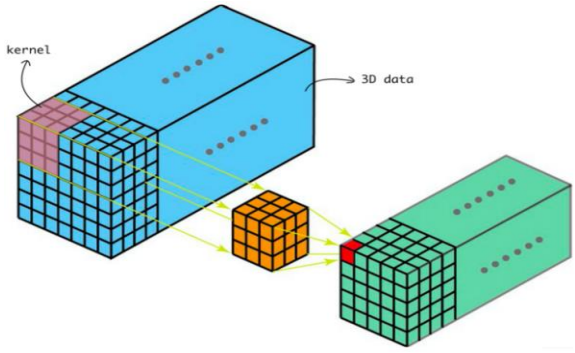

Figure 2.16: 3D Convolution

2.4 Optimizers & Loss functions

A loss function measures the quality of a particular set of parameters by evaluating how well the predicted scores match the ground truth labels in

the training data. There are various types of loss functions, such as Softmax and SVM.

Neural network models learn to map inputs to outputs based on examples, and the choice of the loss function must align with the specific predictive modelling problem, whether it is classification or regression. Additionally, the configuration of the output layer must be suitable for the chosen loss function.

Optimization is the process of finding the set of parameters W that minimizes the loss function. Loss functions and optimizers work together to improve the training of a model by fine-tuning the weights used during training [24].

2.4.1 Gradient Descent

Gradient descent is an optimization algorithm that relies on the first-order derivative of a loss function. It determines the direction in which the weights should be adjusted to reach a minimum. Through backpropagation, the loss is propagated through the layers, and the model's parameters, known as weights, are modified based on the calculated loss to minimize it.

In traditional gradient descent, weights are updated after calculating the gradient across the entire dataset. This means that if the dataset is very large, it may take a long time to converge to the minimum.

2.4.2 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) is an iterative method for optimizing a differentiable objective function. It is a stochastic approximation of gradient descent optimization, where samples are selected randomly (or shuffled) instead of being processed as a single batch or in the order of appearance.

SGD reduces the computational expense of performing typical gradient descent on datasets with a large number of samples.

2.4.3 Mini-Batch Gradient Descent

It is an improvement on both SGD and standard gradient descent. It updates the model parameters after every mini-batch. So, the dataset is divided into various batches and after every batch, the parameters are updated.

Please note the difference in the convergence between the **Stochastic** and the **Mini Batch Gradient Descent** for a better explanation of the smoother convergence of the **Mini Batch Gradient Descent**.

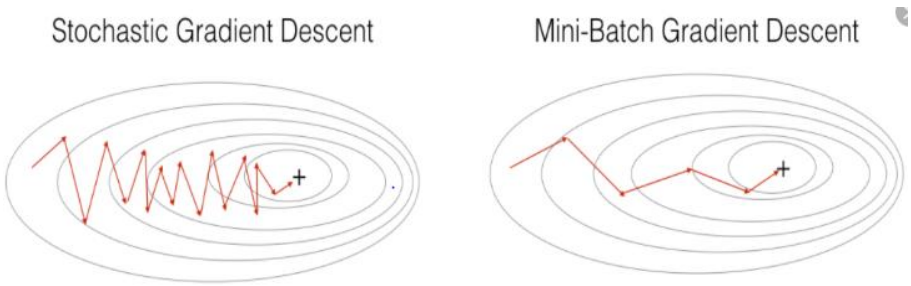

Figure 2.17: Comparison between SGD and mini-batch GD

2.4.4 Momentum

Momentum was invented for reducing high variance in SGD and softening the convergence. It accelerates the convergence towards the relevant

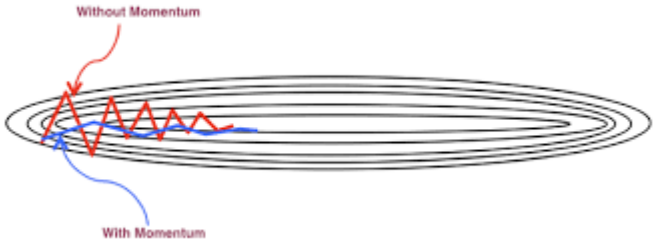

Figure 2.18: Effect of adding Momentum-term

direction and reduces the fluctuation to the irrelevant direction. One more hyperparameter is used in this method known as the momentum term symbolized by 'γ'.

Gradient Descent with momentum considers past gradients to smooth out the updates. It computes an exponentially weighted average of the gradients and uses this average to update the weights. This method typically converges faster than the standard gradient descent algorithm.

2.4.5 RMSprop Optimizer

Gradients of very complex functions, like those in neural networks, tend to either vanish or explode as the data propagates through the function. **RMSprop**, or Root Mean Square Propagation, was developed as a stochastic technique for mini-batch learning to address this issue. It uses a moving average of squared gradients to normalize the gradient. This normalization balances the step size (momentum), decreasing the step for large gradients to avoid exploding and increasing the step for small gradients to avoid vanishing.

Simply put, RMSprop employs an adaptive learning rate rather than treating the learning rate as a fixed hyperparameter. This means that the learning rate changes over time.

2.4.6 ADAM Optimizer

The Adaptive Moment Estimation (ADAM) algorithm combines the heuristics of both Momentum and RMSprop. It computes the exponential average of the gradient and the squared gradient for each parameter. To determine the learning step, the learning rate is multiplied by the average

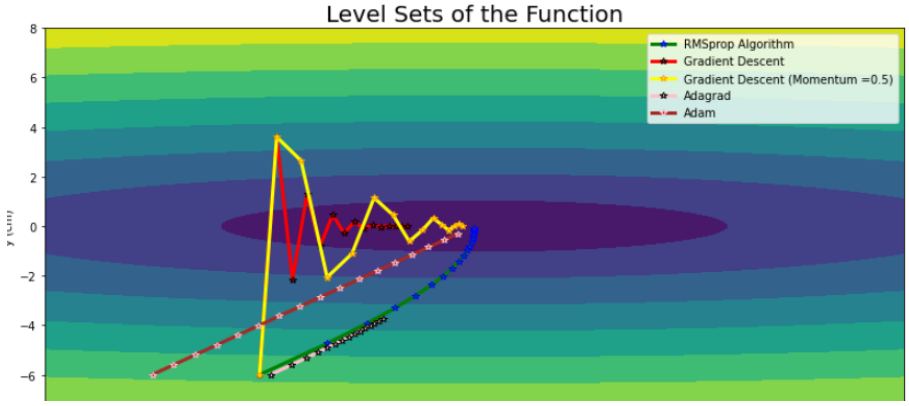

Figure 2.19: Convergence of Different Optimizers

of the gradient (as done in momentum) and divided by the root mean square of the exponential average of the squared gradients (as done in RMSprop). This adjusted step is then used to update the parameters.

2.5 Face Detection

Face detection is a subfield of object recognition specifically focused on identifying and locating human faces within digital images or videos. It plays a crucial role in numerous applications:

- **Facial recognition:** Used for security systems, social media photo tagging, and personalized advertising.
- **Image/video editing:** Enables features like automatic red-eye correction and adding filters to faces.
- **Human-computer interaction (HCI):** Allows systems to track user's gaze or respond to facial expressions in video conferencing.

Techniques for Face Detection:

There are two primary approaches for face detection, each with its own advantages and trade-offs:

2.5.1 Haar Cascade Classifiers

Developed by Viola and Jones, Haar cascades utilize a machine learning technique involving feature extraction and classification. Pre-trained models are readily available in libraries like OpenCV, making implementation straightforward [26].

Advantages:

- Fast and computationally efficient, ideal for real-time applications.
- Easy to use with existing libraries.

Disadvantages:

- **Limited accuracy:** May struggle with variations in facial pose (tilted, up/down) or occlusions (glasses, hair).
- Primarily trained for frontal faces, might miss profiles or non-frontal views.

2.5.2 Multi-task Cascaded Convolutional Networks (MTCNN)

This deep learning-based method leverages a series of interconnected Convolutional Neural Networks (CNNs) for progressively refined face detection. Achieves significantly higher accuracy compared to Haar cascades [25].

Advantages:

- Superior accuracy in handling facial pose variations, occlusions, and non-frontal views.

Disadvantages:

- **Computationally expensive:** Requires more processing power than Haar cascades.
- **More complex setup:** Training and implementing MTCNN might involve additional steps compared to using pre-trained Haar cascades.

Choosing the Right Technique:

The optimal approach for your project depends on your specific requirements:

- If real-time performance is crucial and variations in pose or occlusion are minimal, Haar cascades could be a suitable choice.
- If the application demands high accuracy and robust face detection across diverse scenarios, MTCNN represents a better option, even with its increased processing demands.

2.6 Machine Learning

Machine learning is a subfield of artificial intelligence (AI) that focuses on developing algorithms that can learn from data without being explicitly programmed. These algorithms can then make predictions or decisions on new, unseen data. Machine learning has revolutionized various fields, from self-driving cars and medical diagnosis to recommendation systems and fraud detection.
There are several key paradigms in machine learning, each with its own approach to learning from data [24].

2.6.1 Types

2.6.1.1 Supervised learning

Involves training a model using labeled data, where each data point has a corresponding label or target value. The model learns the relationship between the input features (data points) and the desired output (labels) and can then be used to predict labels for new, unseen data points. Common supervised learning tasks include:

- **Classification:** Predicting a discrete category for a data point (e.g., spam or not spam email).
- **Regression:** Predicting a continuous value for a data point (e.g., housing price prediction).

2.6.1.2 Unsupervised learning

Deals with unlabeled data, where the data points lack predefined labels. The goal is to uncover hidden patterns or structures within the data. Common unsupervised learning tasks include:

- **Clustering:** Grouping similar data points together without predefined categories.
- **Dimensionality reduction:** Reducing the number of features in a dataset while preserving essential information.

2.6.2 Models

2.6.2.1 Support Vector Machines (SVMs)

SVMs are a powerful supervised learning algorithm primarily used for classification tasks. They work by finding a hyperplane (a decision boundary) in high-dimensional space that best separates the data points belonging to different classes [31].

Here's a breakdown of the key concepts in SVMs:

- **Hyperplane:** A linear decision boundary that separates two classes in a two-dimensional space. In higher dimensions, it becomes a hyperplane.
- **Support Vectors:** These are the data points closest to the hyperplane and define the margin between the classes. The SVM aims to maximize this margin to create the best separation between classes.
- **Margin:** The distance between the hyperplane and the closest support vectors from each class. A larger margin indicates a more robust decision boundary and better generalization to unseen data.

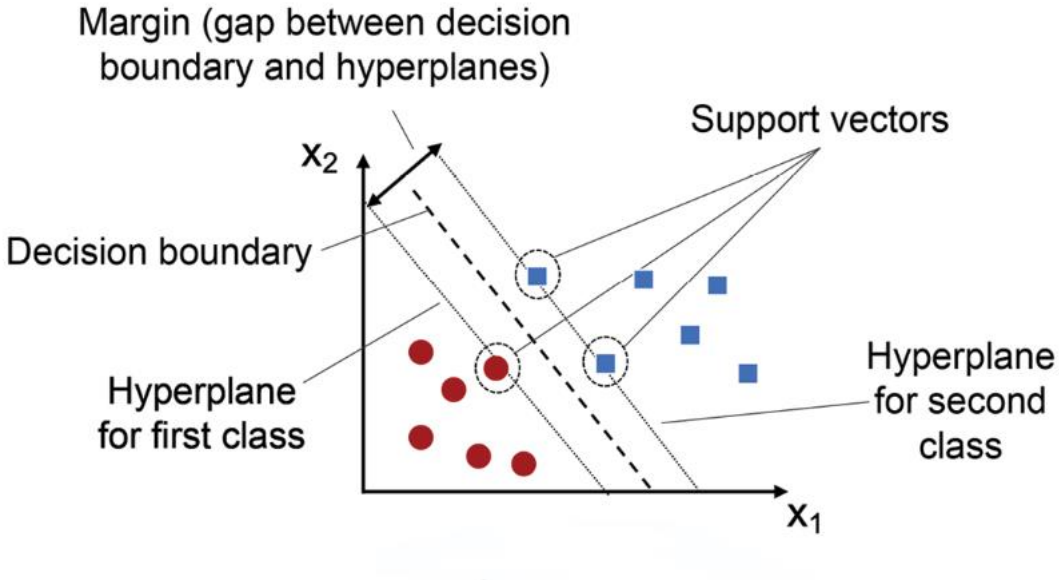

Figure 2.20: SVM Classifier

Advantages of SVMs:

- **High accuracy:** SVMs often achieve high accuracy on classification tasks, especially when dealing with well-separated data sets.
- **Effective for high-dimensional data:** SVMs can handle high-dimensional data efficiently by focusing on the support vectors.
- **Robust to noise:** SVMs are relatively insensitive to outliers and noise in the data.

Disadvantages of SVMs:

- **Kernel trick for non-linear data:** SVMs are primarily designed for linearly separable data. For non-linear data, kernel functions are used to map the data into a higher-dimensional space where it becomes linearly separable. This adds complexity and can be computationally expensive.
- **Black box nature:** SVMs can be challenging to interpret, making it difficult to understand how they arrive at their predictions.
- **Tuning hyperparameters:** SVMs require careful tuning of hyperparameters such as the kernel function and regularization parameters. This can be a time-consuming and expertise-driven process.

SVMs remain a popular and versatile option for classification tasks due to their effectiveness, especially for well-structured datasets. However, for complex and non-linear problems, other algorithms like neural networks might be better suited

2.6.2.2 Decision Trees

Decision trees are a fundamental supervised learning approach for classification and regression tasks. They represent a tree-like structure where each node denotes a feature (characteristic) in the data, and each branch represents a decision based on that feature value. The algorithm iteratively splits the data based on the feature that best separates the classes or minimizes the prediction error [32].

Key Concepts:

- **Root Node:** The starting point of the tree, representing the entire dataset.
- **Internal Nodes:** Nodes containing a question about a specific feature.
- **Leaf Nodes:** Terminal nodes representing the predicted class (classification) or value (regression).
- **Splitting Criteria:** Metrics used to choose the best feature for splitting, like entropy (classification) or mean squared error (regression).

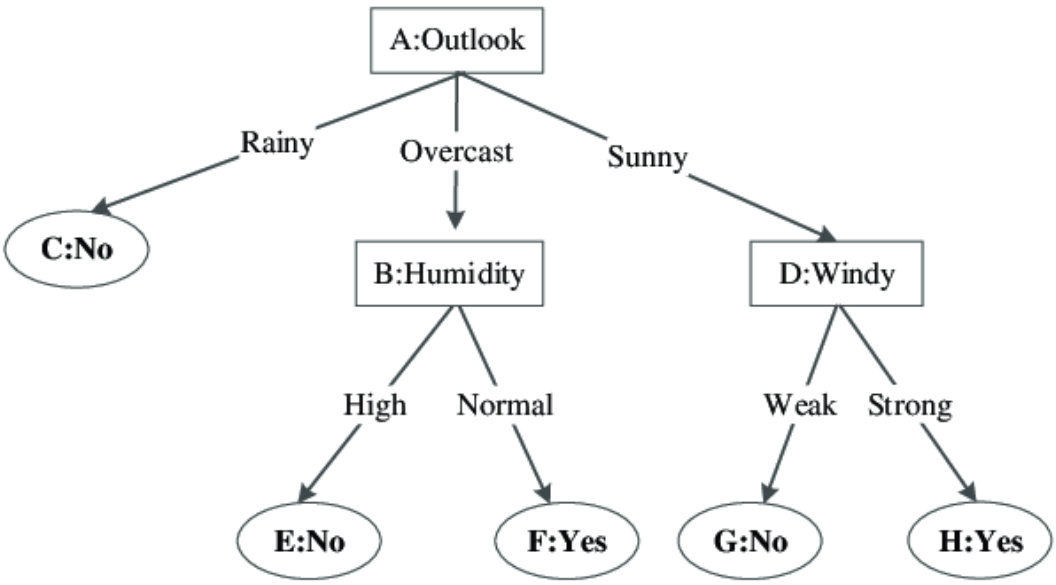

Figure 2.21: Decision Tree Classifier

Advantages of Decision Trees:

- **Interpretable:** Easy to understand how the model makes decisions based on the tree structure.
- **No need for feature scaling:** Handles both numerical and categorical features without scaling.
- **Robust to outliers:** Less susceptible to outliers in the data compared to some other models.

Disadvantages of Decision Trees:

- **Prone to overfitting:** Can become complex and overfit the training data if not carefully controlled.
- **High variance:** Sensitive to small changes in the data, leading to potential instability in predictions.

Decision trees offer a clear decision-making process and work well for various tasks. However, their susceptibility to overfitting and variance requires careful parameter tuning or techniques like pruning or ensemble methods.

2.6.2.3 Random Forest

Random forests address some limitations of decision trees by leveraging the power of ensemble learning. They combine multiple decision trees trained on random subsets of features and/or data points, leading to a more robust and accurate predictor [33].

Core Idea:

- Build multiple decision trees using:
    - **Bootstrapping:** Randomly sampling data points with replacement from the original dataset (creating sub-datasets).
    - **Feature Randomness:** Randomly selecting a subset of features at each split point in the tree (reducing overfitting).
- Combine predictions from all trees by majority vote (classification) or averaging (regression) to make the final prediction.

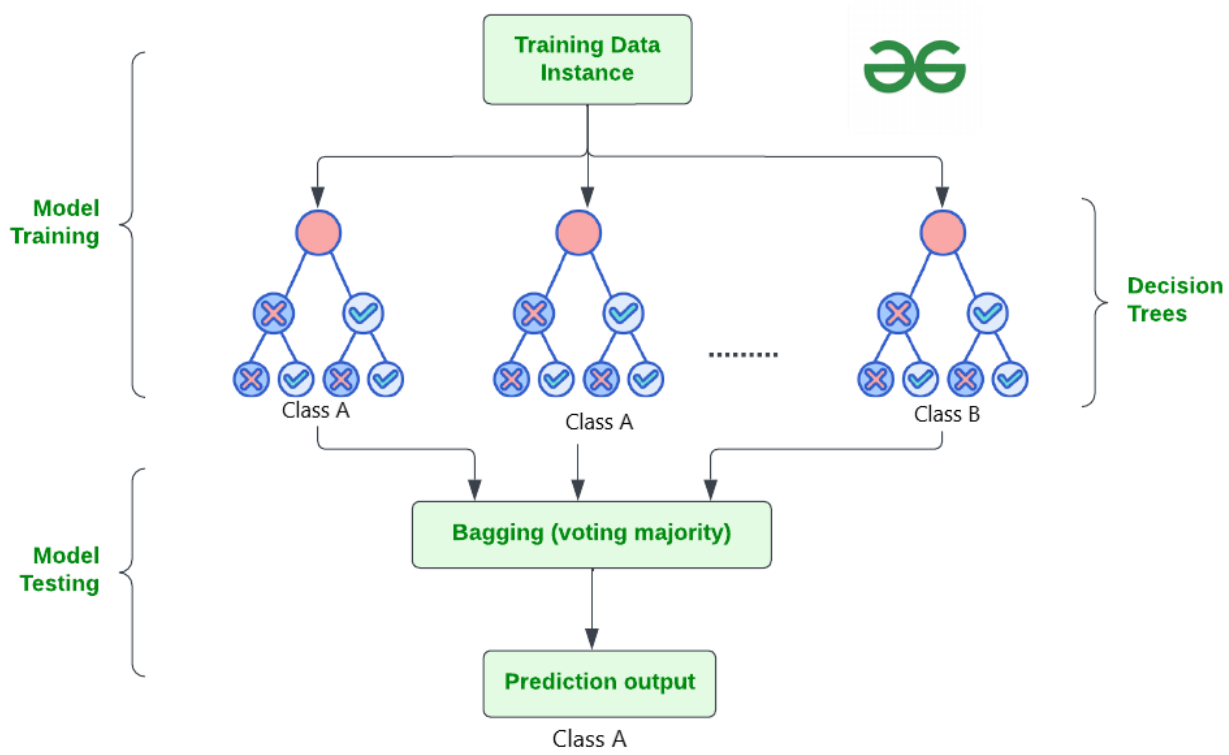

Figure 2.22: Random Forest Classifier

Advantages of Random Forests:

- **Improved accuracy and stability:** Achieves higher accuracy than a single decision tree due to ensemble averaging and reduces variance.
- **Handles missing data:** Can automatically handle missing data points during tree building.
- **Robust to overfitting:** Less prone to overfitting compared to individual decision trees.

Disadvantages of Random Forests:

- **Computationally expensive:** Training multiple trees can be computationally intensive.
- **Black box nature:** While individual trees are interpretable, the overall forest is less so due to the ensemble nature.

- **Tuning hyperparameters:** Requires tuning parameters like the number of trees and the number of features at each split.

Random forests provide a powerful technique for classification and regression tasks, offering increased accuracy and stability compared to individual decision trees. Their ability to handle missing data and reduce overfitting makes them valuable tools in various applications.

2.6.2.4 Stochastic Gradient Descent (SGD) Classifier

The stochastic gradient descent (SGD) classifier is a linear classification algorithm that utilizes the gradient descent optimization technique. It iteratively updates the model's weights (parameters) to minimize the loss function (error) between the predicted and actual labels [35].

Working Mechanism:

- The model predicts class probabilities for each data point.
- The loss function calculates the difference between these predictions and the actual labels.
- The gradient descent algorithm iteratively adjusts the weights based on the negative gradient of the loss function, gradually moving the model towards better predictions.
- Stochastic refers to updating weights using a single data point or a small batch of data points in each iteration, making it efficient for large datasets.

Advantages of SGD Classifiers:

- **Scalability:** Handles large datasets efficiently due to stochastic updates.
- **Simplicity:** Easy to implement and understand the underlying principle.
- **Adaptability:** Can be combined with different loss functions and regularization techniques for diverse tasks.

Disadvantages of SGD Classifiers:

- **Convergence rate:** Can be slow to converge on the optimal solution, especially for complex datasets

- **Sensitivity to hyperparameters:** Requires careful tuning of learning rate and other hyperparameters to achieve optimal performance.
- **Potential for local minima:** May get stuck in local minima of the loss function, leading to suboptimal solutions.

2.6.2.5 XGBoost Classifier

XGBoost (Extreme Gradient Boosting) is a powerful and widely used ensemble method for classification and regression tasks. It builds upon the concept of gradient boosting, iteratively adding new decision trees to the ensemble while focusing on improving predictions for previously misclassified data points [34].

Key Features:

- **Gradient Boosting:** Each new tree in the ensemble focuses on correcting errors from previous trees, leading to a more accurate overall model.
- **Regularization:** Techniques like L1/L2 regularization are incorporated to prevent overfitting and improve model generalizability.
- **Sparsity:** XGBoost encourages sparse trees (trees with fewer splits), improving efficiency and interpretability.
- **Parallelization:** Designed for efficient parallelization across multiple cores or GPUs, speeding up training on large datasets.

Advantages of XGBoost Classifiers:

- **High accuracy and efficiency:** Often achieves state-of-the-art performance on various classification benchmarks.
- **Scalability:** Handles large datasets efficiently due to parallelization and sparse models.
- **Flexibility:** Can handle different loss functions and data types, making it adaptable to diverse tasks.

Disadvantages of XGBoost Classifiers:

- **Complexity:** Tuning hyperparameters can be more involved compared to simpler models.
- **Black box nature:** While individual trees are interpretable, the overall XGBoost model is less so due to the ensemble nature.

- **Computationally expensive:** Training can be computationally intensive, especially for large datasets with complex hyperparameter tuning.

2.7 NLP (Natural Language Processing)

Natural Language Processing (NLP) is a subfield of artificial intelligence concerned with the interactions between computers and human language. It encompasses various techniques for enabling computers to understand, process, and generate human language. NLP has numerous applications in fields like machine translation, sentiment analysis, text summarization, chatbots, and information retrieval [36].

2.7.1 TF-IDF (Term Frequency-Inverse Document Frequency)

TF-IDF is a numerical weighting scheme commonly used in NLP to represent the importance of a word within a document or collection of documents. It aims to highlight words that are frequent within a specific document but rare across the entire corpus (collection).

Key Components:

- **Term Frequency (TF):** Measures how often a word appears in a document divided by the total number of words in the document.
- **Inverse Document Frequency (IDF):** Measures how often a word appears across all documents in the corpus, with rarer words having higher IDF scores.

The TF-IDF score for a word is calculated by multiplying the TF and IDF values. Words with high TF-IDF scores are considered more relevant and informative for that specific document compared to the entire collection.

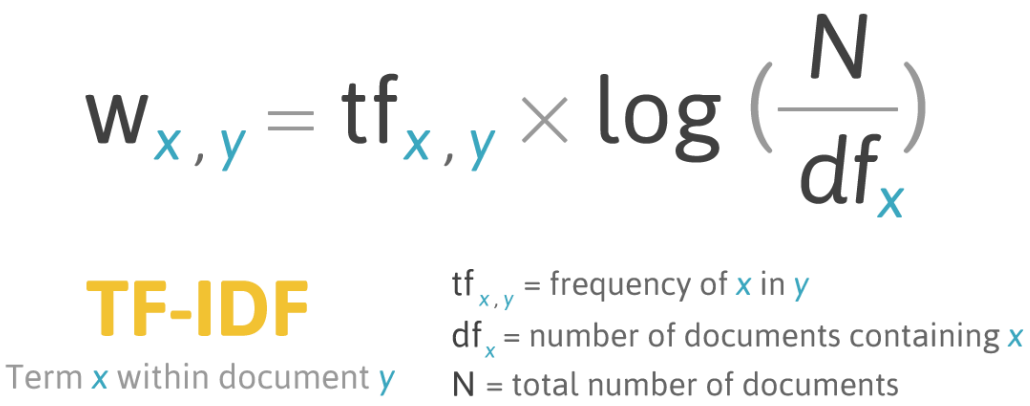

Figure 2.23: TF-IDF

Advantages of TF-IDF:

- **Simple and efficient:** Easy to compute and understand.
- **Effective for document weighting:** Useful for identifying keywords and summarizing document content.
- **Robust to document length:** Handles documents of varying lengths effectively.

Disadvantages of TF-IDF:

- **Limited semantic understanding:** Does not capture word relationships or context.
- **Sensitivity to rare words:** Can be skewed by very rare words with high IDF but low relevance.
- **Not ideal for short texts:** Less effective for short documents like tweets or emails.

2.7.2 Count Vectorizer

The Count Vectorizer is a common tool in NLP used to convert a collection of text documents into a numerical matrix, suitable for machine learning algorithms. It works by counting the occurrences of each word (feature) in each document.

Function:

- Takes a collection of text documents as input.

- Analyses each document and identifies unique words (vocabulary).
- Creates a matrix where each row represents a document and each column represents a unique word.
- Fills the matrix with the count of each word's occurrence in the corresponding document.

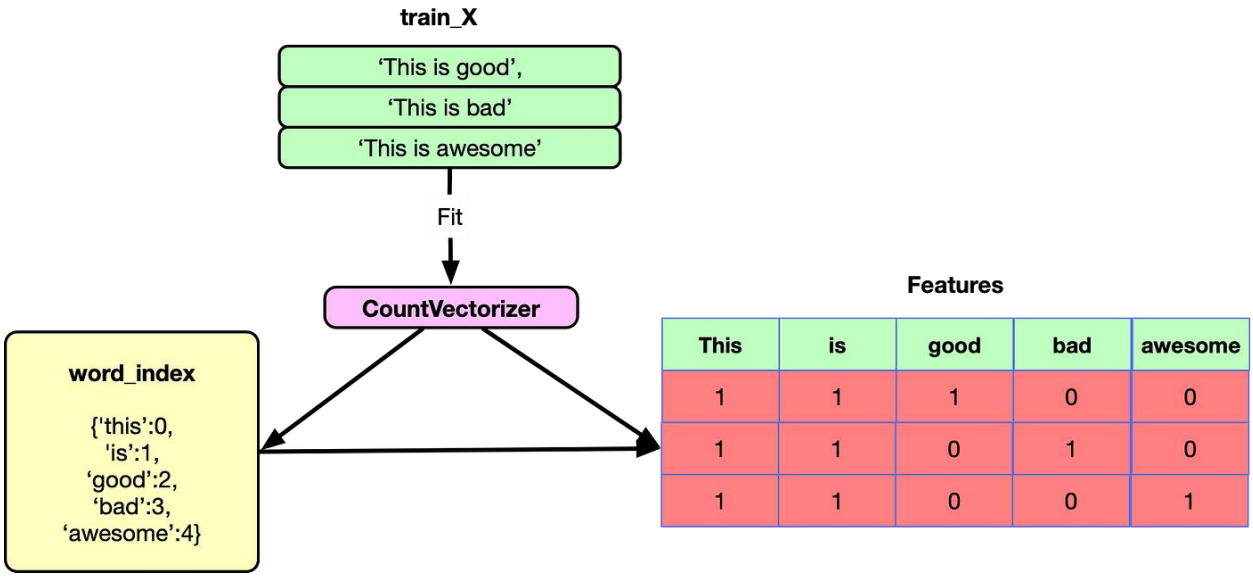

Figure 2.24: Count Vectorizer

Advantages of Count Vectorizer:

- **Simple and efficient:** Easy to implement and understand.
- **Suitable for various tasks:** Useful for tasks like document classification and topic modelling.
- **Preserves word order (optional):** Can optionally retain word order information for specific NLP tasks.

Disadvantages of Count Vectorizer:

- **Limited semantic understanding:** Ignores word relationships and context.
- **High dimensionality:** Can create high-dimensional matrices for large vocabularies, impacting efficiency.
- **Sensitivity to rare words:** Can be skewed by very rare words with low information content.

2.7.3 GloVe (Global Vectors for Word Representation)

GloVe (Global Vectors for Word Representation) is a popular word embedding technique in NLP that captures semantic relationships between words. It learns vector representations for words based on their co-occurrence statistics within a large corpus of text data. These word vectors encode semantic similarities, enabling tasks like word analogies, synonym identification, and sentiment analysis [37].

Core Idea:

- Analyzes the co-occurrence patterns of words within a large text corpus.
- Constructs a co-occurrence matrix where rows and columns represent words, and each cell represents the frequency of words appearing together within a specific window size.
- Applies matrix factorization techniques to decompose the co-occurrence matrix and extract word vectors.
- These word vectors capture semantic relationships, where similar words have more similar vector representations in the high-dimensional space.

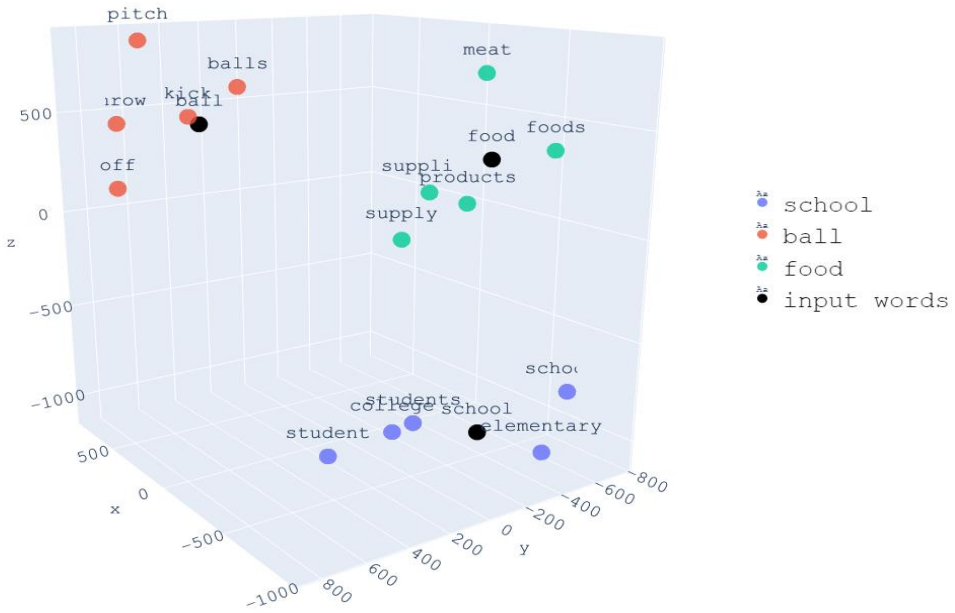

Figure 2.25: GloVe

Advantages of GloVe:

- **Captures semantic relationships:** Represents words based on their context and semantic meaning.
- **Handles polysemy:** Can handle words with multiple meanings by capturing context-dependent representations.
- **Efficient for large datasets:** Scales well to massive text corpora for learning meaningful word embeddings.

Disadvantages of GloVe:

- **Computational cost:** Training on large corpora can be computationally expensive.
- **Black box nature:** The internal workings of the matrix factorization process are not easily interpretable.
- **Limited syntactic understanding:** Does not explicitly capture word order or grammar.

2.8 Audio Signals

2.8.1 MFCCs (Mel-frequency cepstral coefficients)

MFCCs are a feature extraction technique for audio signals, mimicking human hearing. They capture the "shape" of the sound's frequency content, focusing on how our ears perceive different frequencies. This makes them robust to noise and speaker variations [27].

Process:

1. Divide the audio signal into short segments (frames).
2. Analyse the frequency content of each frame.
3. Apply a mel scale to emphasize lower frequencies like our ears do.
4. Extract features (MFCCs) that capture the overall shape of the frequency spectrum.

Benefits:

- Robust to noise
- Speaker-independent
- Efficient representation of audio data

Applications:

- Speech recognition (e.g., voice assistants)
- Speaker identification
- Music information retrieval

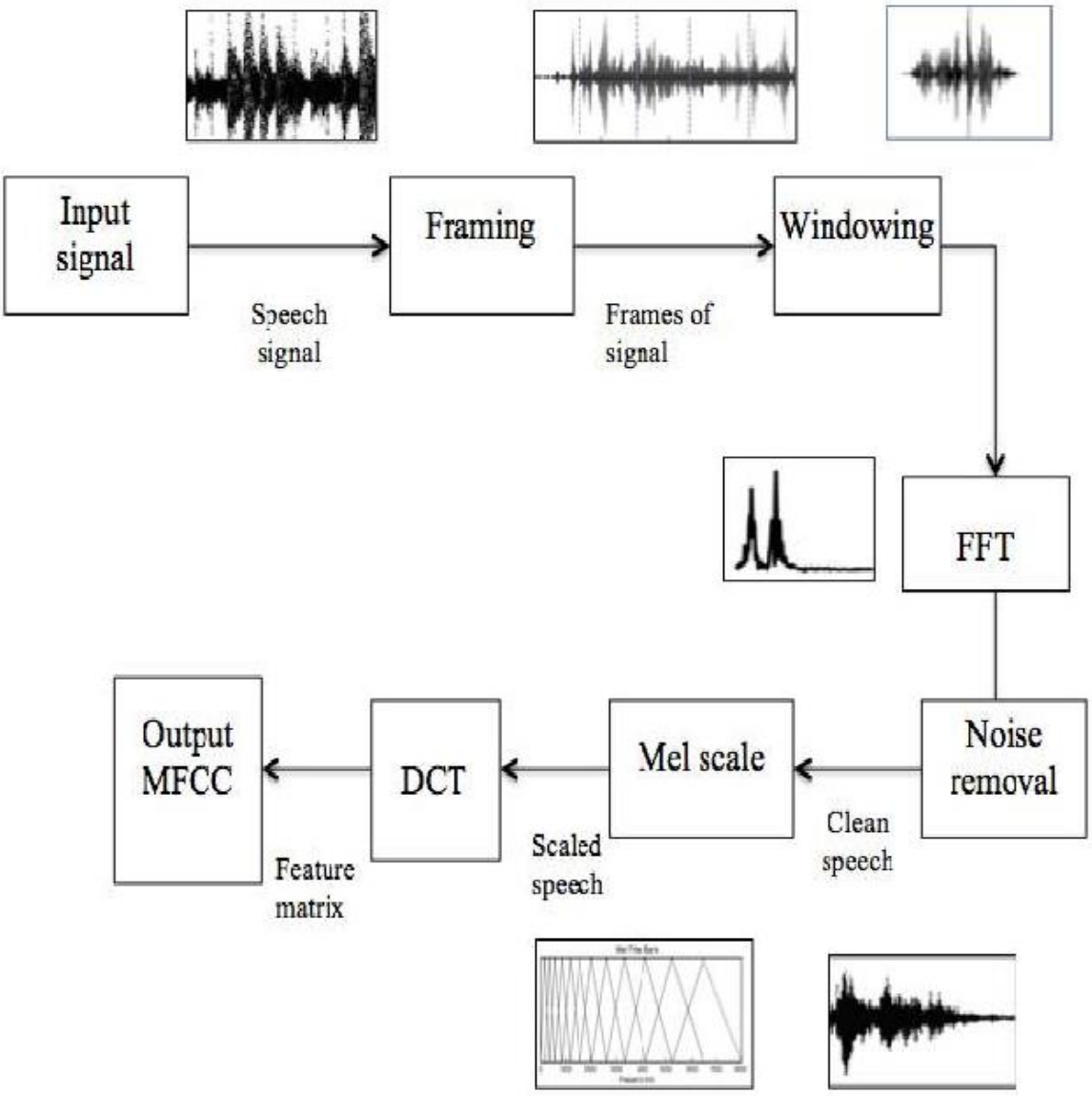

Figure 2.26: MFCC steps

2.9 Micro Expressions

Micro expressions are fleeting, involuntary facial expressions that briefly reveal a person's underlying emotions. They can last for fractions of a sec-

ond, often masked by attempts to conceal true feelings. Understanding micro expressions can provide valuable insights into someone's emotional state, potentially revealing hidden thoughts or disagreements [2].

2.9.1 Facial Action Units (AUs) and Coding Systems

To analyse micro expressions effectively, researchers and practitioners utilize facial action unit (AU) coding systems. These systems provide a standardized method for identifying and classifying the basic building blocks of facial movement. A popular system is the Facial Action Coding System (FACS) [28]

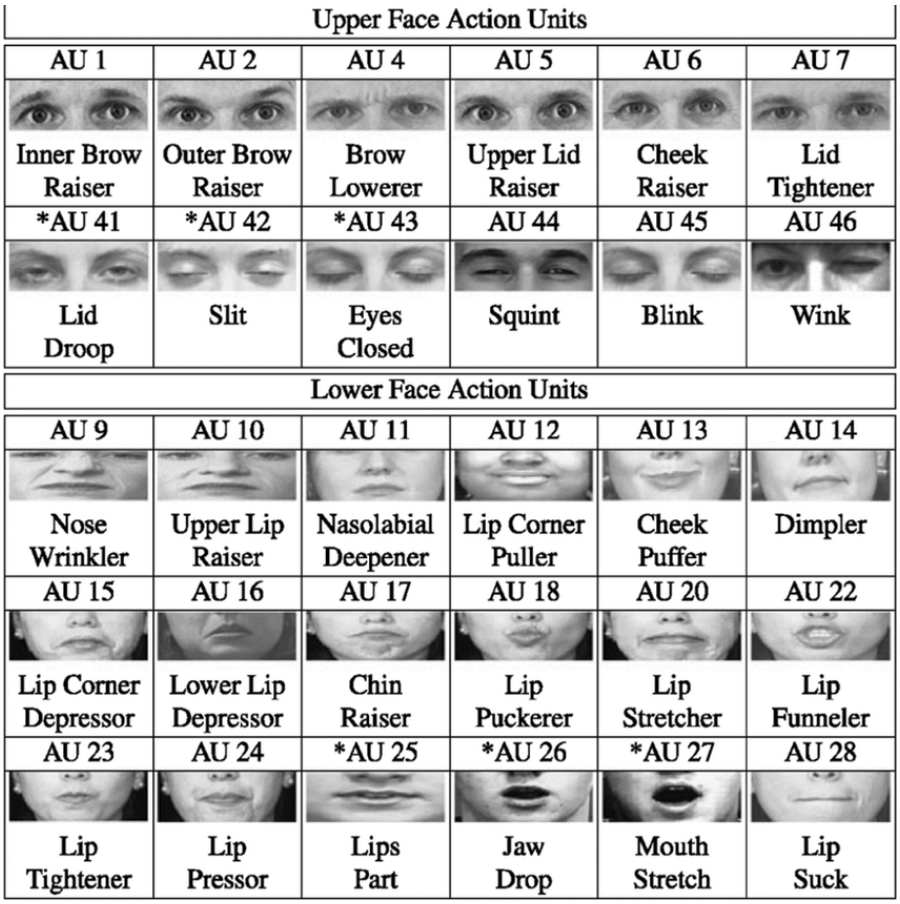

Figure 2.27: Facial Action Coding System

FACS defines a set of action units (AUs) that correspond to specific muscle movements on the face. Trained coders analyze facial expressions and assign AU codes to identify the underlying emotions. Analyzing micro expressions often involves detecting combinations of AUs associated with specific emotions.

2.10 Multimodal Data Fusion

In many real-world scenarios, information about a phenomenon or system can be acquired from various sources, each providing a distinct perspective. Multimodal data fusion leverages these different data modalities (e.g., images, text, audio, sensor data) to create a more comprehensive and robust representation for tasks like classification, prediction, or decision-making. By combining information from multiple modalities, fusion techniques aim to outperform models trained solely on individual data types.

2.10.1 Early Fusion

Early fusion, also known as feature-level fusion, combines the raw features extracted from each modality before feeding them into a single machine learning model. This allows the model to learn the relationships between features from different modalities directly [29].

Advantages:

- **Potential for capturing complex interactions:** Enables the model to learn how features from different modalities influence each other, potentially leading to improved performance.
- **Simpler model training:** Requires training only one machine learning model on the fused feature representation.

Disadvantages:

- **High dimensionality:** Can lead to high-dimensional feature spaces, potentially increasing computational demands and the risk of overfitting.
- **Sensitivity to feature scaling:** Requires careful scaling of features from different modalities to ensure they contribute equally during learning.

2.10.2 Late fusion

Late fusion, also known as decision-level fusion, combines the predictions or decisions made by individual models trained on each modality separately [29]. Common late fusion techniques include:

- **Majority Voting:** Assigns the final prediction to the class that receives the most votes from individual models. Simple and effective, but may not leverage complementary information from different modalities.
- **Weighted Average:** Combines predictions from individual models based on pre-defined weights reflecting their perceived reliability or importance. Requires careful weight tuning for optimal performance.

Advantages:

- **Reduced dimensionality:** Uses the lower-dimensional outputs from individual models, potentially reducing computational costs.
- **Flexibility with models:** Allows using different models for each modality, potentially leveraging specialized algorithms suited for specific data types.

Disadvantages:

- **May not capture complex interactions:** Limits the model's ability to learn relationships between features from different modalities.
- **Requires well-performing individual models:** Relies on the accuracy of predictions from each modality.

Figure 2.28: Early Fusion vs Late Fusion

2.11 Related Work

2.11.1 Multimodal Deception Detection Using Real-Life Trial Data [5]

In their study, Şen et al. (2022) developed a multimodal deception detection system using real-life trial data, focusing on verbal, acoustic, and visual modalities. They utilized a dataset of videos from public court trials, extracting features from linguistic, visual, and acoustic channels. The final classifier combined these modalities via score-level classification, achieving an accuracy of 83.05% in subject-level deceit detection. They employed a combination of manually annotated and automatically extracted features, comparing their system's performance against human capabilities. The results demonstrated that their system outperformed average non-expert human ability in identifying deception.

2.11.2 Lie Detection using Speech Processing Techniques [8]

In the study by Fathima Bareeda et al. (2021), the authors proposed a non-invasive lie detection technique using speech processing. They utilized the "Real life trial data" dataset, which contains audio clips of truthful and de-

ceptive statements. The methodology involved preprocessing the audio signals to remove noise using the Short-Time Fourier Transform (STFT) based thresholding technique. Key features, particularly Mel-frequency cepstral coefficients (MFCCs), were extracted from the preprocessed signals. These features were used to train a Support Vector Machine (SVM) classifier to distinguish between truthful and deceptive speech. The system achieved an accuracy of 81% using polynomial kernels, demonstrating the effectiveness of their approach in leveraging speech features for lie detection.

2.11.3 Deception Detection in Videos using the Facial Action Coding System [9]

In their study, Ahmed et al. (2021) developed a deception detection system utilizing the Facial Action Coding System (FACS) to analyze facial expressions in videos. The researchers extracted facial action units (AUs) using the open-source software OpenFace, which provided over 700 facial features, of which 32 were related to AUs. They used a long short-term memory (LSTM) model to analyze the temporal sequences of these features for detecting deception. The dataset included real-life trial videos, which were preprocessed to ensure the focus was on the confessors' faces and relevant frames.

The authors divided the videos into chunks of 30 frames, balancing the dataset with equal numbers of truthful and deceptive chunks. They trained the LSTM model using these chunks, optimizing the model by including a single LSTM layer, a dropout layer to prevent overfitting, and a dense layer for the final output. The system achieved a correct classification rate (CCR) of up to 89.49%, outperforming several other visual-only deception detection approaches. Additionally, the study explored cross-dataset validation using three datasets, revealing challenges in combining datasets with different natures and stakes.

2.11.4 Introducing Representations of Facial Affect in Automated Multimodal Deception Detection [10]

Mathur and Matarić (2020) introduced a novel approach for automated multimodal deception detection by incorporating dimensional representations of facial affect: valence and arousal. The study utilized a video dataset of courtroom testimonies where individuals communicated truthfully or deceptively. Leveraging the Aff-WildNet, a state-of-the-art deep neural network trained on the Aff-Wild database, the researchers extracted continuous representations of facial valence and arousal.

They combined these affective features with other visual, vocal, and verbal cues to train Support Vector Machines (SVMs) and experimented with unimodal and multimodal fusion methods. Unimodal models trained solely on facial affect achieved an AUC of 80%. The highest-performing multimodal approach, which combined facial affect with visual and vocal features using adaptive boosting, achieved an AUC of 91%.

The study's results highlighted the discriminative power of facial affect in deception detection, supporting existing psychological theories on the relationship between affect and deception. This research underscores the potential of affect-aware systems for improving automated detection of deception and other social behaviours in real-world scenarios.

2.11.5 A Deep Learning Approach for Multimodal Deception Detection [11]

Krishnamurthy et al. (2018) developed a multimodal neural model for deception detection, utilizing features from video, audio, text, and micro-expressions. They extracted visual features using a 3D Convolutional Neural Network (3D-CNN) that captured spatiotemporal aspects of facial expressions. Textual features were derived from video transcripts using a Convolutional Neural Network (CNN) with pre-trained Word2Vec embeddings. Audio features were extracted using the openSMILE toolkit, focusing on high-dimensional acoustic features. Micro-expressions were manually annotated and included as binary features.

The researchers employed a Multi-Layer Perceptron (MLP) model for classification, using rectified linear unit (ReLU) activation functions and

dropout for regularization. They explored different data fusion techniques, including concatenation and Hadamard product, to combine features from various modalities. The model was trained using the cross-entropy loss function and optimized with stochastic gradient descent.

Experimental results on a dataset of real-life courtroom trial videos demonstrated the effectiveness of their approach. The proposed model achieved an accuracy of 96.14% and an ROC-AUC of 0.9799, outperforming existing state-of-the-art techniques. Their findings highlighted the significance of multimodal feature extraction in accurately detecting deceptive behaviour.

2.11.6 Constructing Robust Emotional State-based Feature with a Novel Voting Scheme for Multi-modal Deception Detection in Videos [20]

Yang et al. (2022) introduced a multimodal deception detection framework that constructs robust emotional state-based features using a novel voting scheme. The study addresses the data scarcity problem in deception detection by employing several preprocessing methods to handle unusable frames in video data. Visual features were extracted using deep learning methods to recognize emotional states from facial expressions, while audio features were extracted using the openSMILE toolkit to capture emotional cues from speech.

A novel emotional state transformation (EST) feature was developed by combining emotional states from both visual and audio modalities. The EST feature considers information from both spatial and temporal dimensions, enhancing the accuracy of deception detection. A voting method was implemented to integrate emotional state information from both modalities, improving the robustness of the final emotional state determination.

The framework was evaluated on a dataset of real-life courtroom trial videos, demonstrating significant improvements in accuracy and ROC-AUC compared to state-of-the-art methods. The proposed methods achieved an accuracy of 92.78% and an ROC-AUC of 0.9265, highlighting the effectiveness of incorporating multimodal emotional state features for deception detection.

3. Analysis and Design

3.1 System Overview

We introduce the entire architecture of our system (*see figure 3.1*). The system utilizes various technologies to achieve multimodal fusion of different data types.

3.1.1 System Architecture

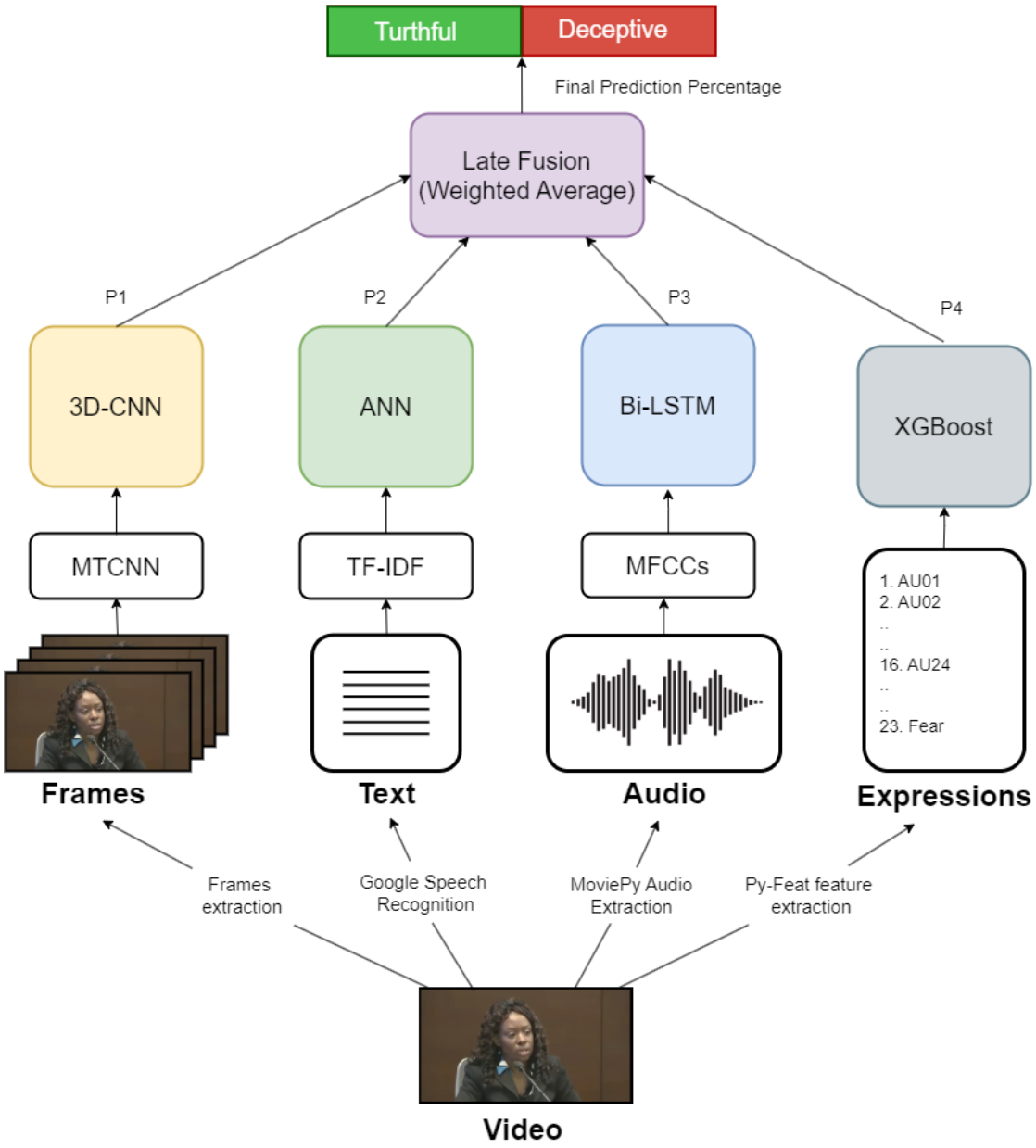

Figure 3.1: System Architecture

First, the input to the system is a complete video. The separate modalities are processed individually. For videos, frames are extracted, and faces are detected before being fed into a 3D-CNN model. For text, TF-IDF is used for feature extraction, and these features are then fed into a simple ANN. For audio, MFCCs are extracted and fed into a Bi-LSTM model. Additionally, micro-expressions are extracted using the Py-Feat library (see section 5.2.3). Finally, the outputs from these different models are fused using a weighted average approach to aggregate the various insights, resulting in a prediction percentage indicating whether the input is truthful or deceptive.

3.1.2 System Users

- Intended Users:

  The system is built for the public: the user uploads a video to the system to detect whether a subject is lying or not while giving a statement.
- User Characteristics
  - Familiar with website applications/browsing.
  - Familiar with English.

3.2 System Analysis & Design

3.2.1 Use Case Diagram

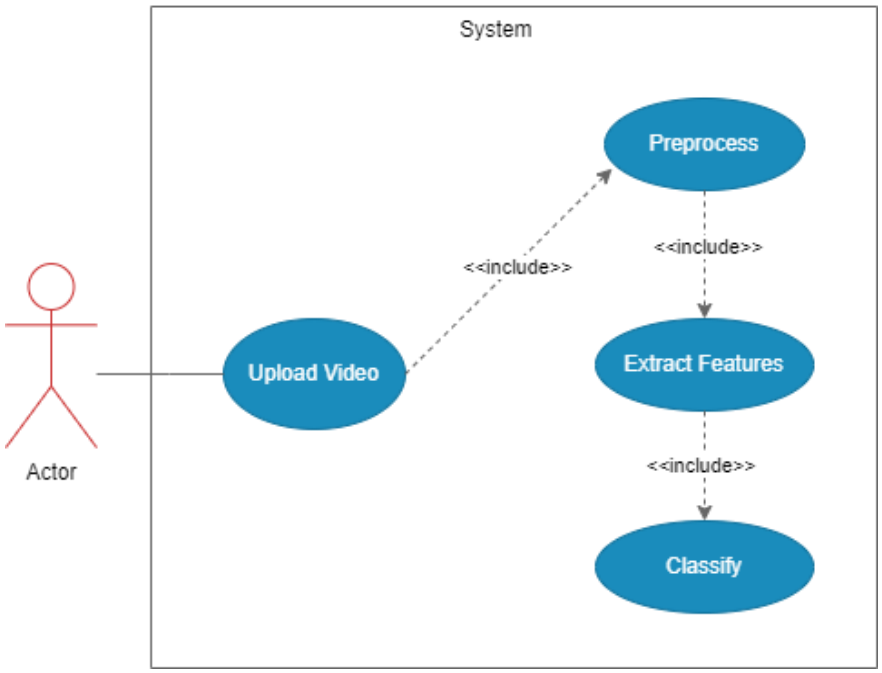

Figure 3.2: Use Case Diagram

3.2.1.2 Description of Use Cases

1. Upload Video
    - Description: Takes a video as an input parameter and starts the classification process.
    - User: Actor.
2. Preprocessing
    - Description: Preprocesses the different modalities individually.
    - User: System.
    - Expected Output: Cleaned data in each modality.
3. Extract Features
    - Description: Extracts the desired features from each modality.
    - User: System.
    - Expected Output: A vector of the extracted desired features.
4. Classify
    - Description: Classifies the resulting features into either truthful or deceptive
    - User: System.
    - Expected Output: A percentage indicating the probability that the user is lying.

3.2.2 Flow of Events

- The user uploads a video to the website.
- The website saves the video to the backend and calls the classification module.
- The module processes the input video and outputs a classification percentage.
- The classification result is sent to the website.
- The website displays the result to the user.

3.2.3 Sequence Diagram

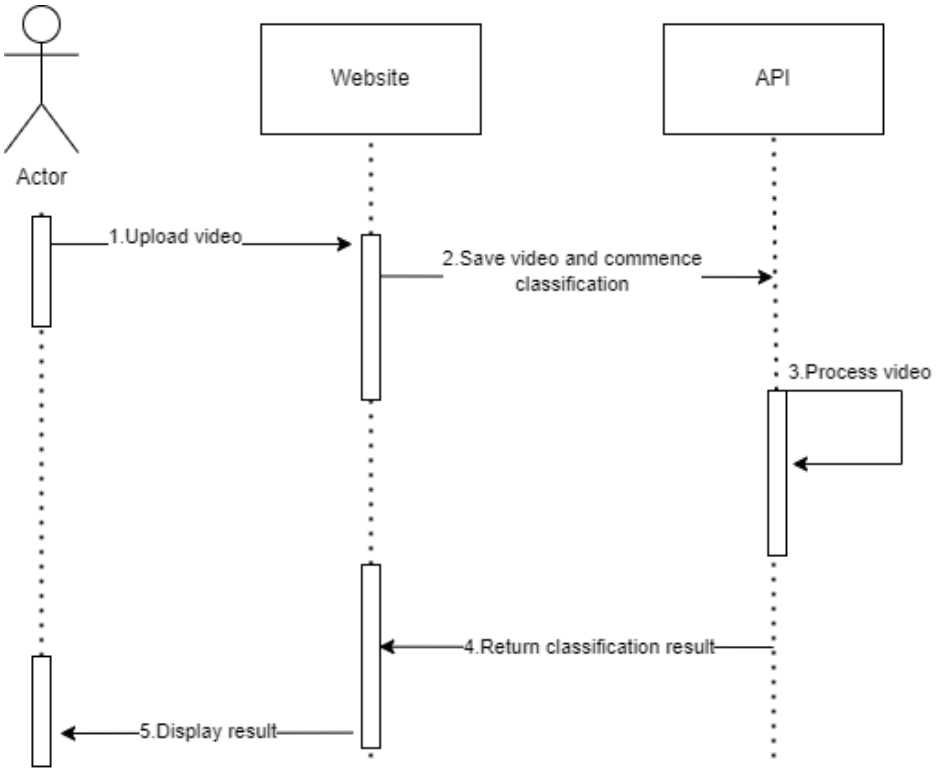

Figure 3.3: Sequence Diagram

4. Dataset

To evaluate our deception detection model, we utilized a real-life deception detection dataset as referenced in [3]. This dataset comprises 121 video clips of courtroom trials, with 61 clips classified as lies and the remaining 60 classified as truth.

The dataset is annotated with micro-expressions identified in each video. An accompanying Excel sheet includes columns for 39 possible micro-expressions (see Figure 4.2). However, we also extracted additional micro-expressions beyond those provided in the dataset (see Section 5.2.2). Additionally, the dataset includes transcriptions for each video (see Figure 4.3).

Some videos in the dataset had issues, such as multiple subjects in the frame simultaneously and dialogues between two people. These problems were addressed by manually cropping and clipping the videos, as no automated solutions were available. Additionally, a few videos were severely corrupted and unsuitable for training, so they were omitted from our testing.

The dataset also has bias issues. The same subjects often reappear in multiple videos, exhibiting consistent behaviour, which can introduce bias. Moreover, there is a disproportionate gender representation in deceptive videos, which can further skew results if the data is split randomly. To mitigate this, we manually split the dataset during our experiments to ensure that the training and test sets did not have overlapping subjects.

Figure 4.1: Dataset screenshots

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1	id	OtherGest	Smile	Laugh	Scowl	otherEyeb	Frown	Raise	OtherEyeN	Close-R	X-Open	Close-BE	gazeInterl	gazeDowr	gazeUp	otherGaze	gazeSide	openMout	closeMou	lipsDown	lipsUp	lipsRetrac	lipsProtru	SideTurn
2	trial_lie_001.mp4	1	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	1	1	0	0	0	
3	trial_lie_002.mp4	1	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	1	0	1	0	0	0	
4	trial_lie_003.mp4	1	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	1	1	0	0	0	
5	trial_lie_004.mp4	1	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0	0	0	
6	trial_lie_005.mp4	1	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	1	0	1	0	0	0	
7	trial_lie_006.mp4	1	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	1	1	0	0	0	
8	trial_lie_007.mp4	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	1	1	0	0	0	
9	trial_lie_008.mp4	1	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0	1	0	0	0	
10	trial_lie_009.mp4	1	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	1	1	0	0	0	
11	trial_lie_010.mp4	0	1	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0	
12	trial_lie_011.mp4	0	0	0	1	0	0	1	0	0	0	1	0	1	0	0	0	0	1	1	0	0	0	
13	trial_lie_012.mp4	1	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	1	1	0	0	0	
14	trial_lie_013.mp4	1	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	1	1	0	0	0	
15	trial_lie_014.mp4	1	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0	0	1	0	0	
16	trial_lie_015.mp4	1	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	1	0	1	0	0	
17	trial_lie_016.mp4	1	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	1	0	0	1	0	0	
18	trial_lie_017.mp4	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	1	1	0	0	0	
19	trial_lie_018.mp4	1	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0	1	0	0	0	
20	trial_lie_019.mp4	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	1	0	0	1	0	0	
21	trial_lie_020.mp4	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	1	0	0	1	0	0	
22	trial_lie_021.mp4	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	1	1	0	0	0	
23	trial_lie_022.mp4	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	1	0	0	1	0	0	
24	trial_lie_023.mp4	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	1	0	0	1	0	0	

Figure 4.2: Micro expressions annotations

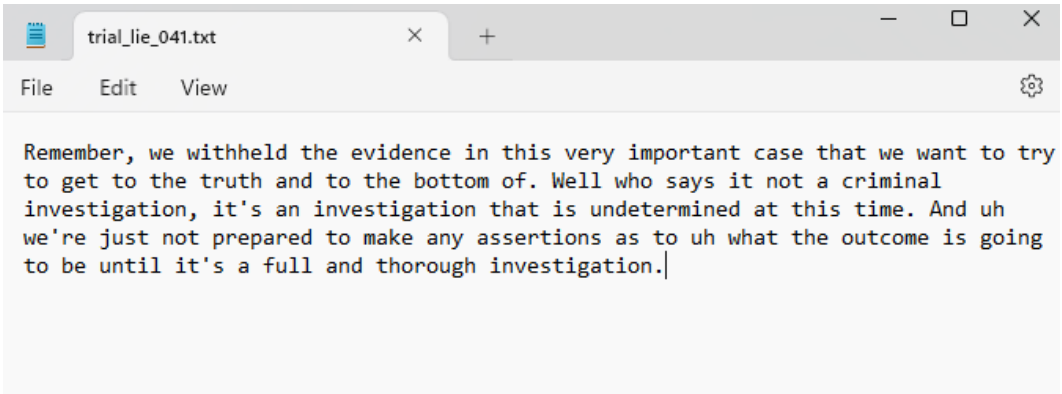

Figure 4.3: Transcriptions for video

5. Implementation

5.1 Environment Setup & Tools

For this project, we primarily utilized Python, which is currently the most popular programming language in recent academic and industry publications. Python was chosen for its user-friendly nature and its extensive libraries, many of which are specifically designed for building machine learning and deep learning models.

Most of our scripts were executed in Visual Studio Code (VS Code). For version control and collaborative work, we relied on GitHub to manage and integrate our contributions effectively.

5.1.1 Environments

1. Localhost
    - While developing our scripts for preprocessing, training, and testing, we ran them locally on a GPU-enabled machine with sufficient computational power for our tasks. This setup negated the need for external platforms such as Google Colab or Kaggle. Additionally, we used our local setup to develop our final React.js website and to integrate our models.

Table 1: Environment Specifications

CPU	Intel Core i7 9750H
RAM	16 GB
GPU	RTX 2060

5.1.2 Packages and Libraries

1. React
    - A free and open-source front-end JavaScript library used to create user interfaces, particularly single-page applications, based on UI components.
2. Os

- This module provides a way of using operating system-dependent functionality in a portable manner within Python.

3. OpenCV (cv2)
    - OpenCV (Open Source Computer Vision Library) is an open-source software library for computer vision and machine learning. It was developed to facilitate the integration of machine perception in commercial products and provide a common infrastructure for computer vision applications.
4. Pandas
    - Pandas is a fast, flexible, and easy-to-use open-source data analysis and manipulation tool built on top of the Python programming language.
5. TensorFlow
    - TensorFlow is an open-source framework developed by Google for running machine learning, deep learning, and other statistical and predictive analytics workloads.
6. Scikit-Learn
    - Scikit-learn is a crucial library for the Python programming language, typically used in machine learning projects. It focuses on providing machine learning tools, including mathematical, statistical, and general-purpose algorithms, forming the basis for many machine learning technologies. It is a free tool that is highly valuable in algorithm development for machine learning and related technologies.
7. NLTK
    - NLTK (Natural Language Toolkit) is a Python library for natural language processing (NLP) tasks such as tokenization, stemming, tagging, parsing, and classification. It provides easy-to-use interfaces to linguistic data and resources, making it a popular choice for NLP research and application development.
8. Django
    - Django is a high-level Python web framework that simplifies the development of web applications by providing built-in fea-

tures such as an ORM (Object-Relational Mapping) for database interaction, a powerful templating engine, and an admin interface.

9. Py-Feat
    - Py-Feat provides a comprehensive set of tools and models to easily detect facial expressions (Action Units, emotions, facial landmarks) from images and videos. It also offers functionality to preprocess, analyse, and visualize facial expression data.

5.2 Data Preprocessing

5.2.1 Video Frames Preprocessing

In our approach, we used each video sample in its entirety. Other approaches, such as those described in [9], [19], split each video sample into multiple segments of a specified length. While this segmentation method addresses the issue of the limited number of available videos by creating more samples, it introduces inaccuracies because it is unclear in which part of the video the deception occurs. Therefore, we chose to downsample the videos to achieve a uniform sequence length and to use each testimony as a single input for our models during both training and testing.

Video frames were preprocessed using the following steps:

1. Downsampling
    - Video frames were downsampled using frame skipping to achieve a consistent number of frames in each video.
2. Face Extraction
    - Faces were extracted using a face detector (MTCNN or Haar Cascade Detector) as the background was unnecessary for our deception detection task.
3. Frames Resizing
    - The extracted face frames were resized to ensure uniform width and height across all video samples.
4. Converting RGB to Grayscale

- This step was optional, but it was beneficial in reducing the input size and improving model efficiency.

5. Normalization
    - The video colour channels were normalized by dividing by 255 to scale the values to a range of 0 to 1 for better convergence.

Results:

Figure 5.1: Input frame

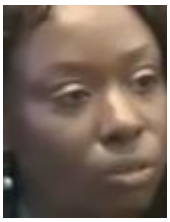

Figure 5.2: Frame after cropping the face

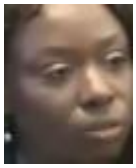

Figure 5.3: Frame after resizing

Figure 5.4: Frame after optional conversion to greyscale

5.2.2 Micro Expressions Extraction

Although the dataset contains manually annotated expressions that can be useful for deception detection, we opted to extract our own annotations using the Py-Feat [22] library to make the system fully automated.

Various signals and facial features were extracted from videos using Py-Feat [22], focusing on Action Units (AUs) and emotions, which are important for deception detection. Features were extracted every 30 frames to balance computational efficiency and detail. The resulting records were then combined into a single mean feature vector.

Py-Feat [22] allows us to choose from various models for the face, landmarks, action units, and emotion. Our decision for every model is as follows:

- Face model => Retina Face
- Landmark Model => Mobilefacenet
- Action units Model => XGB
- Emotion Model => Resmasknet

5.2.3 Text preprocessing

The transcribed text underwent preprocessing as part of an NLP pipeline. The steps were as follows:

1. Punctuation
    - Punctuation was removed to standardize the text.
2. Lowercasing
    - Words were converted to lowercase for consistency.
3. Tokenization
    - The text was split into individual words.
4. Stop Word Removal
    - Stop words were filtered out to enhance text quality.
5. Lemmatization
    - Words were reduced to their base forms.

6. Vectorization
    - The cleaned text was transformed using a vectorizer such as TF-IDF or GloVe, enabling numerical representation suitable for machine learning and deep learning models.

5.2.4 Audio Preprocessing

In this audio modality, we extracted the Mel-frequency cepstral coefficients (MFCC) from the audio signals as they can aid in revealing deceptive patterns as seen in the previous research [8].

The preprocessing pipeline for audio involved the following steps:

1. Audio Extraction
    - Extracted audio from the video.
2. Noise Reduction
    - Performed Short-Time Fourier Transform (STFT) noise reduction to clean the audio.
3. MFCC Extraction
    - Extracted Mel-frequency cepstral coefficients (MFCC) features at specified framing windows.
    - We used a frame length of 0.025 seconds and a hop length of 0.01 seconds at a target sampling rate of 44100 Hz.

    Additionally, due to the varying durations of the videos, the number of MFCC segments produced was inconsistent. To address this, we downsampled the MFCC segments to unify the input shape for our sequential models.

5.3 Models Implementation

5.3.1 Video Frames Modality

Lie detection via video leverages involuntary facial expressions, eye movements, and body gestures potentially indicative of deception, previous research in [9], [10], [11], [19] shows that neural networks based on CNN architectures were successful in capturing patterns associated with deception.

5.3.1.1. Deep Learning

In this modality we mainly experimented with two neural network architectures which are:

1. CNN-LSTM
    - This architecture combines spatial features from 2D convolutions with temporal features extracted from an LSTM layer, then it feeds these features to a few dense layers and finally, there is a sigmoid layer at the end for final classification.

Model architecture:

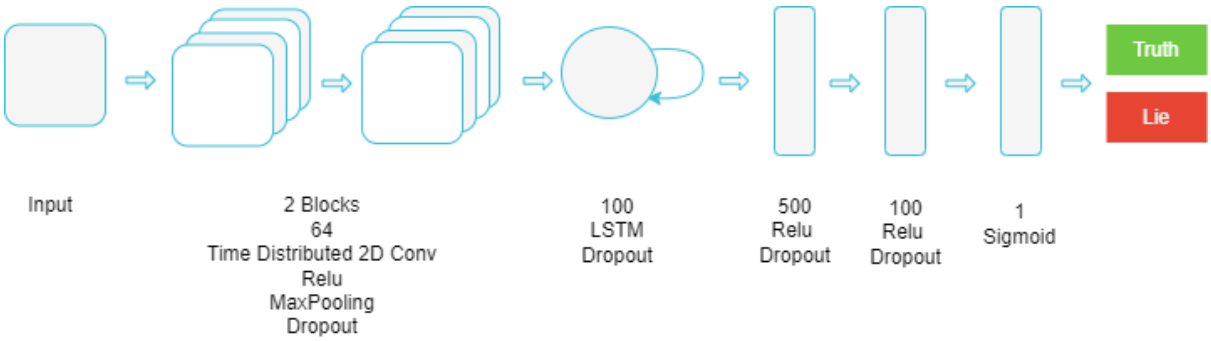

Figure 5.5: CNN-LSTM Model Architecture

2. 3D CNN
    - This architecture leverages 3D convolutions to extract temporal and spatial features, these features are then fed to multiple dense layers and at last, a sigmoid layer is used for the final classification.

Model architecture:

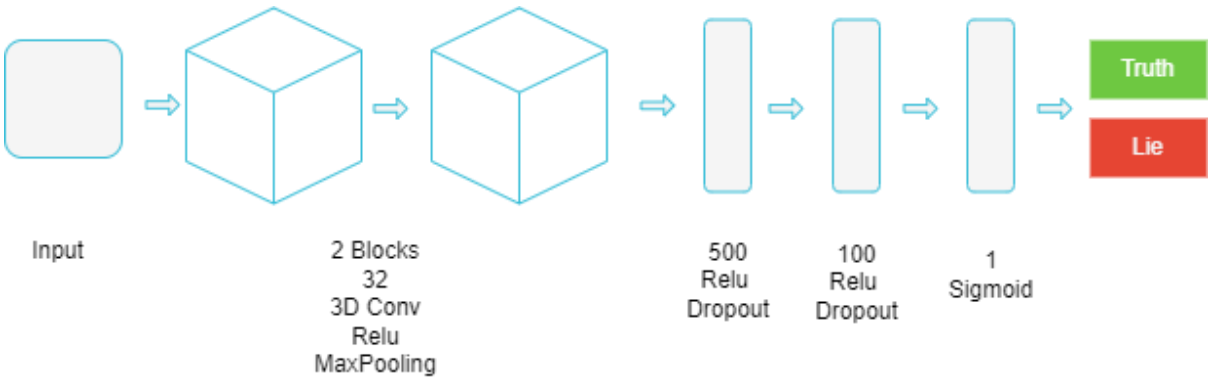

Figure 5.6: 3D CNN Model Architecture

5.3.2 Micro Expressions Modality

After extracting the mean Action Units and emotions feature vector from each video sample using Py-Feat in the preprocessing step (see section 5.2.2), we employed machine learning classifiers to identify deceptive cues in these micro expressions. Our approach is supported by research in [3], [5], which demonstrates the effectiveness of machine learning classifiers in detecting deception from micro expressions. We chose machine learning classifiers over deep learning neural networks because this data modality is less complex, additionally, the limited number of samples makes neural networks more prone to overfitting.

5.3.2.2 Machine Learning

We experimented with machine learning algorithms such as:

- Support Vector Machine (SVM).
- Decision Tree.
- Random Forest.
- Extreme Gradient Boosting (XGBoost).

5.3.3 Text Modality

Vectorized textual data was used to detect deceptive patterns, employing various machine learning algorithms for this task. Additionally, we used simple deep learning models, such as a basic Artificial Neural Network (ANN). Although Recurrent Neural Networks (RNNs) and Transformers are state-of-the-art for text classification, they were unsuitable for our task due to the limited number of samples. These complex networks are prone to overfitting or may fail to learn the patterns effectively with insufficient data.

5.3.3.1 Machine Learning

We employed machine learning algorithms such as:

- Support Vector Machine (SVM).
- Stochastic Gradient Descent (SGD)
- Extreme Gradient Boosting (XGBoost).

5.3.3.2 Deep Learning

We tried simple ANN architectures:

- This architecture consists of only two hidden layers with a small number of perceptions in each layer, finally, there is one sigmoid layer to get the final prediction.

Model Architecture:

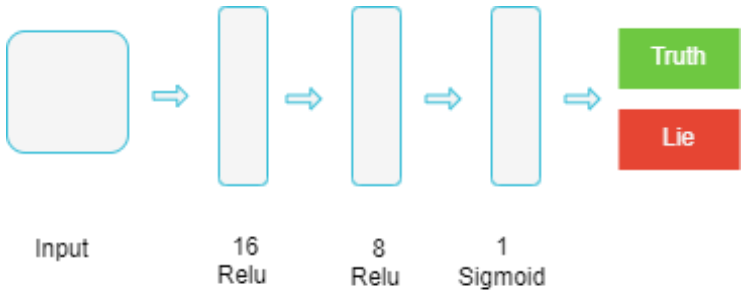

Figure 5.7: ANN Model Architecture

5.3.4 Audio Modality

The extracted MFCCs were utilized to analyse subtle vocal cues such as pitch, tone, and speech pace. We employed Recurrent Neural Networks (RNNs) to analyse the previously extracted MFCC sequences (see section 5.2.4) to detect deception.

5.3.4.1 Deep Learning

We experimented with simple RNN based architectures:

- This architecture has one layer that uses a type of RNNs such as GRUs, LSTMs and Bidirectional LSTMs, followed by one sigmoid layer for final classification.

Model architecture:

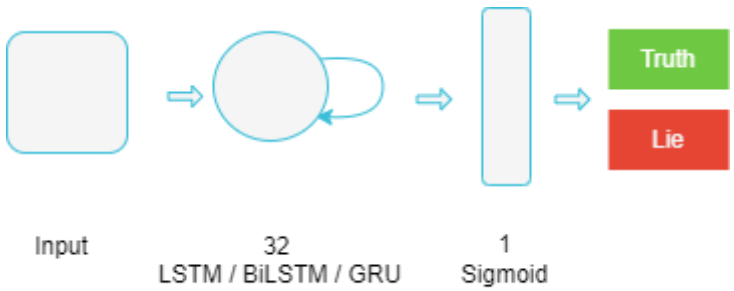

Figure 5.8: RNN Model Architecture

5.4 Website, and Model Integration

We used React for the frontend to build the website's UI components. For the backend, we chose Django as it provides seamless integration between the frontend and our models. Django, built in Python -the same language used for preprocessing and models training- allowed us to maintain consistency without needing to adjust our implementations.

Web Application Architecture:

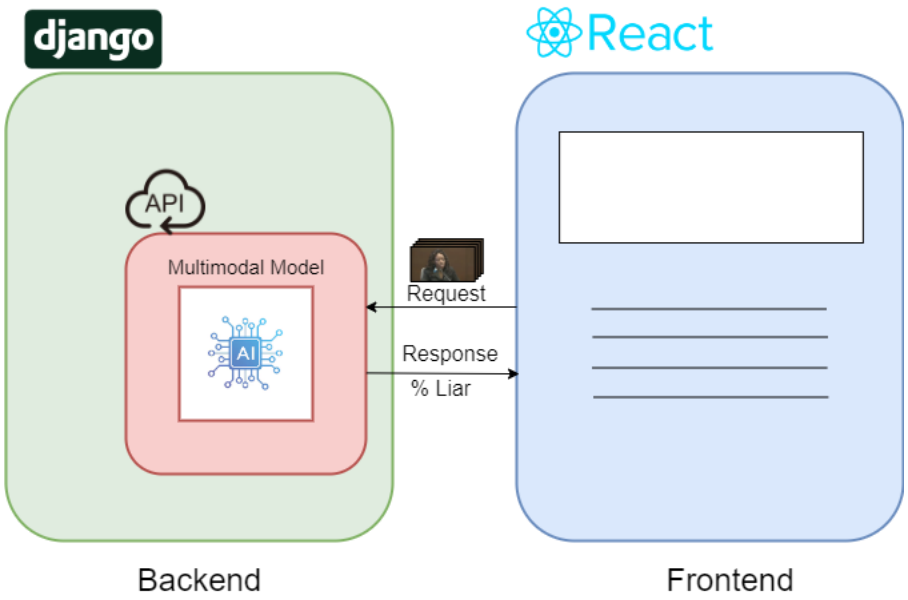

Figure 5.9: Web Application Architecture

6. Results

6.1 Training and Testing Datasets

The dataset used in our testing (see section 4) is highly prone to bias issues and personality bleeding across training and testing sets if split automatically, because there is unequal distribution of genders and subjects across the two classes, in addition to some subjects mostly exhibiting one behaviour (i.e. mostly lying or mostly telling truth in their respective video samples).

Considering ethical implications and our main objective, which is to make a reliable unbiased system, we opted to manually split the dataset such that the testing and training sets have no overlapping subjects, in addition to an equal gender distribution.

Some previous research ignored this issue with the dataset such as [8], [9], [19]. This is problematic because an automatic split of training and testing set will result in falsely high accuracies, this is because models that rely on visual or acoustic feature are likely to classify anything such as specific persons or genders rather than detecting deception and still achieve a reasonably high accuracy given the dataset distribution issues.

Other research [5], [11] acknowledged this issue and they used K-Fold cross validation based on subjects rather than videos to overcome this issue. However, we chose to manually pick 20% of the videos for the testing set to mitigate the bias issues.

6.2 Results Summary

In this section, we will delve into the detailed work conducted for each data modality. The tables below present the experiments performed on each modality, highlighting the parameters used and the highest accuracy achieved by each model on its respective data modality.

6.2.1 Video Frames Modality Results

In our experiments with the video modality, we experimented with different sequence lengths and image sizes in addition to different face detectors across our 3D CNN and CNN-LSTM neural network architectures (see section 5.3.1.1).

RMSProp optimizer with default learning rate gave consistently better results than any other optimizer or learning rate so it was used in all the following recordings, additionally checkpoints and early stopping were used to prevent overfitting and to save the model with the best metrics and the following results were obtained:

Table 2: Video Modality Results

Deep Learning Models			
Model Name	Face Detector	Sample Shape	Accuracy
3D CNN	Haar Cascade Frontal Face	100x64x64x1	83.3%
3D CNN	Haar Cascade Frontal Face	100x96x96x1	79.1%
3D CNN	Haar Cascade Frontal Face	500x64x64x1	79.1%
3D CNN	**MTCNN**	**100x64x64x1**	**87.5%**
3D CNN	MTCNN	100x96x96x1	87.5%
3D CNN	MTCNN	500x64x64x1	83.3%

CNN-LSTM	Haar Cascade Frontal Face	100x64x64x1	70.8%
CNN-LSTM	Haar Cascade Frontal Face	100x96x96x1	75%
CNN-LSTM	Haar Cascade Frontal Face	500x64x64x1	66.6%
CNN-LSTM	MTCNN	100x64x64x1	79.1%
CNN-LSTM	MTCNN	100x96x96x1	75%
CNN-LSTM	MTCNN	500x64x64x1	70%

6.2.2 Expressions Modality Results

Different machine learning classifiers were tested with different settings on the extracted micro expressions feature vectors (see section 5.2.2) and the most prominent results were recorded in the following table:

Table 3: Expressions Modality Results

Machine Learning Models		
Model Name	Parameters	Accuracy
SVM	kernel = ‘poly’, degree = 2, C = 1, gamma = ‘auto	54.1%
SVM	kernel = ‘linear, C = 10, gamma = ‘auto	54.1%

Decision Tree	criterion = ‘gini’, Splitter = ‘best’, Max_depth = None	62.5%
Decision Tree	criterion = ‘gini’, Splitter = ‘best’, Max_depth = 6	54.1%
Random For-est	n_estimators = 1500, max_leaf_nodes = 10, max_depth = 50	66.7%
Random For-est	n_estimators = 3000, max_leaf_nodes = 10, max_depth =50	54.1%
XGBoost	**max_depth = 6, learning_rate = 0.3, n_estimators = 100, booster = ‘gbtree’, gamma = 0**	**75%**
XGBoost	max_depth = 10, learning_rate = 0.3, n_estimators = 1000, booster = ‘gbtree’, gamma = 0	70.8%

6.2.3 Text Modality Results

Different combinations of text vectorizers were crossed with various machine learning classifiers as well as a simple neural network and the most important results were recorded in the table below:

Table 4: Text Modality Results

Deep Learning Models			
Model Name	Vectorizer	Parameters	Accuracy

ANN	GloVe 100D	learning_rate = 0.01, optimizer = ‘adam’, epochs = 10	70.8%
ANN	GloVe 100D	learning_rate = 0.001 optimizer = ‘adam’, epochs = 20	58.3%
ANN	**TF-IDF**	**learning_rate = 0.01 optimizer = ‘adam’, epochs = 10**	**79.1%**
ANN	TF-IDF	learning_rate = 0.001 optimizer = ‘adam’, epochs = 20	54.1%
Machine Learning Models			
Model Name	Vectorizer	Parameters	Test Accuracy
SGD	TF-IDF	Alpha = 0.0001, learning_rate = 'optimal	70.8%
SGD	Glove 100D	Alpha = 0.0001, learning_rate = 'optimal	54.1%
SVM	TF-IDF	kernel = 'linear', C = 10	75%
SVM	TF-IDF	kernel = rbf, C = 10	66.7%
SVM	GloVe 100D	kernel = linear, C = 10	54.1%
XGBoost	TF-IDF	max_depth = 6, learning_rate = 0.3, n_estimators = 100,	66.7%

XGBoost	GloVe 100D	max_depth = 6, learning_rate = 0.3, n_estimators = 100,	62.5%

6.2.4 Audio Modality Results

Experiments were done on previously extracted MFCC sequences (see section 5.2.4) with different sequence lengths, and sequences were fed to different RNN based architectures (see section 5.3.4).

Similar to the video modality, checkpoints and early stopping were used to prevent over fitting and the results were as follows:

Table 5: Audio Modality Results

Deep Learning Models			
Model Name	Number of MFCCs samples	Parameters	Accuracy
GRU	700	learning_rate = 0.0001 optimizer = ‘adam’, epochs = 40	70.8%
LSTM	2000	learning_rate = 0.0001 optimizer = ‘adam’, epochs = 40	66.7%
LSTM	700	learning_rate = 0.0001 optimizer = ‘adam’, epochs = 40	70.8%
LSTM	2000	learning_rate = 0.0001 optimizer = ‘adam’, epochs = 40	66.7%

Bi-LSTM	**700**	**learning_rate = 0.0001 optimizer = 'adam', epochs = 40**	**79.1%**
Bi-LSTM	700	learning_rate = 0.01 optimizer = 'adam', epochs = 50	62.5%
Bi-LSTM	2000	learning_rate = 0.0001 optimizer = 'adam', epochs = 40	75%

6.2.5 Multimodal Fusion Results

In our attempts to achieve multimodal fusion between the previously mentioned modalities, we explored output level fusion techniques using our best model in each modality such as weighted average and majority voting, the following table showcases our findings:

Table 6: Multimodal Fusion Results

Model Name	Parameters	Accuracy
Late fusion (A + V + T) Weighted average	Weights = [0.33, 0.33, 0.33]	87.5%
Late fusion (A + V + T) Weighted average	Weights = [0.15, 0.5, 0.35]	91.6%

Late fusion (A + V + T) Majority Voting	**_**	**95.8%**
Late fusion (A + V + T+ E) Weighted average	Weights = [0.25, 0.25, 0.25, 0.25]	87.5%
Late fusion (A + V + T + E) Majority Voting	_	91.66%

6.3 Competitive analysis

Comparing our best results with some of the state of the art multimodal approaches results, particularly the results in [5], [9], [10], [11], [20].

The table below highlights the methodologies used in previous works and their respective accuracy and how our approach compares to it.

Our system performs reasonably well in comparison to other approaches, however, the approach provided by Krishnamurthy, Gangeshwar, et al [11] performs a little bit better but that is expected since their system is semi-automatic (it relies on manual annotations for micro expressions) while our system is fully automated.

Table 7: Competitive analysis

Citation	Dataset	Methodology	Best Accuracy
Şen, M. Umut, et al (2020) [5].	-Real Life Deception Detection Dataset -Silesian Deception -Bag of lies	Late fusion with visual, vocal and linguistic features	83%
Ahmed, Hammad Ud Din, et al (2021) [9]	Real Life Deception Detection Dataset	FACS with LSTM	89.4%
Mathur, Leena, and Maja J. Matarić (2020) [10]	Real Life Deception Detection Dataset	Visual and vocal features with AdaBoost	84%
Krishnamurthy, Gangeshwar, et al (2018) [11]	Real Life Deception Detection Dataset	Early fusion with visual. vocal and linguistic features	96.1%
Yang, Jun-Teng, Guei-Ming Liu, and Scott C-H. Huang (2020) [20]	Real Life Deception Detection Dataset	EST, ME and IS13 features with logistic regression	92.7%
Proposed Method	**Real Life Deception Detection Dataset**	**Late fusion with visual, vocal and linguistic features**	**95.8%**

6.4 Drawbacks

Although our approach achieved great results in detecting deception on the Real Life Deception Detection Dataset, it still has a few shortcomings, mainly the following:

1. Our approach is resource-intensive, requiring approximately 2-3 minutes on our system (specified in Table 1) to preprocess and predict a single video sample.

2. The models are trained on a small dataset of 121 videos, which may not ensure optimal performance in real-world scenarios.

3. The models are specifically trained to detect deception in courtroom environments and may not achieve the same level of accuracy when applied to other contexts.

7. User Manual

1. The user should click on the Upload File button.

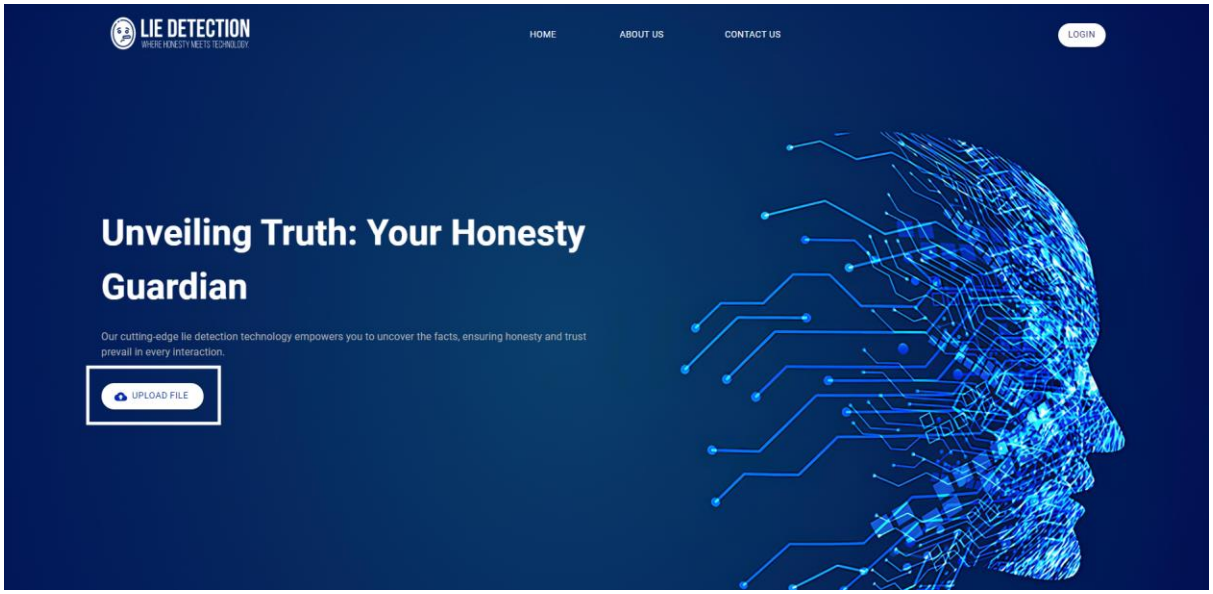

Figure 7.1: Upload File button

2. The user should browse the video that needs to be classified as a lie or a truth.

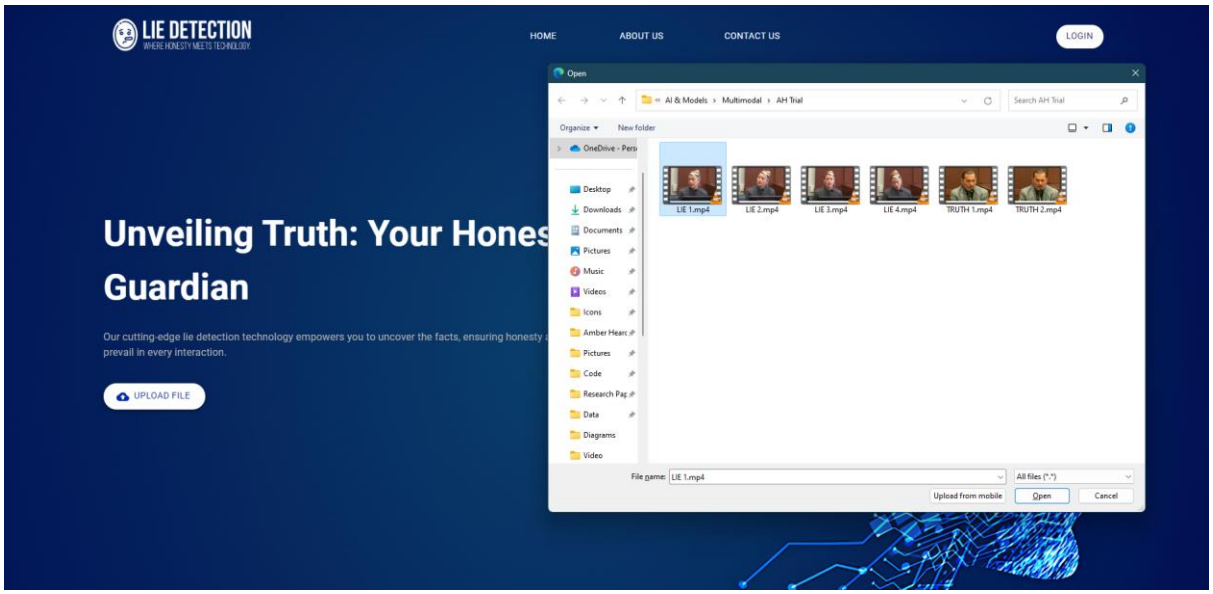

Figure 7.2: Choosing a file

3. The user should wait for the result to be computed through the API.

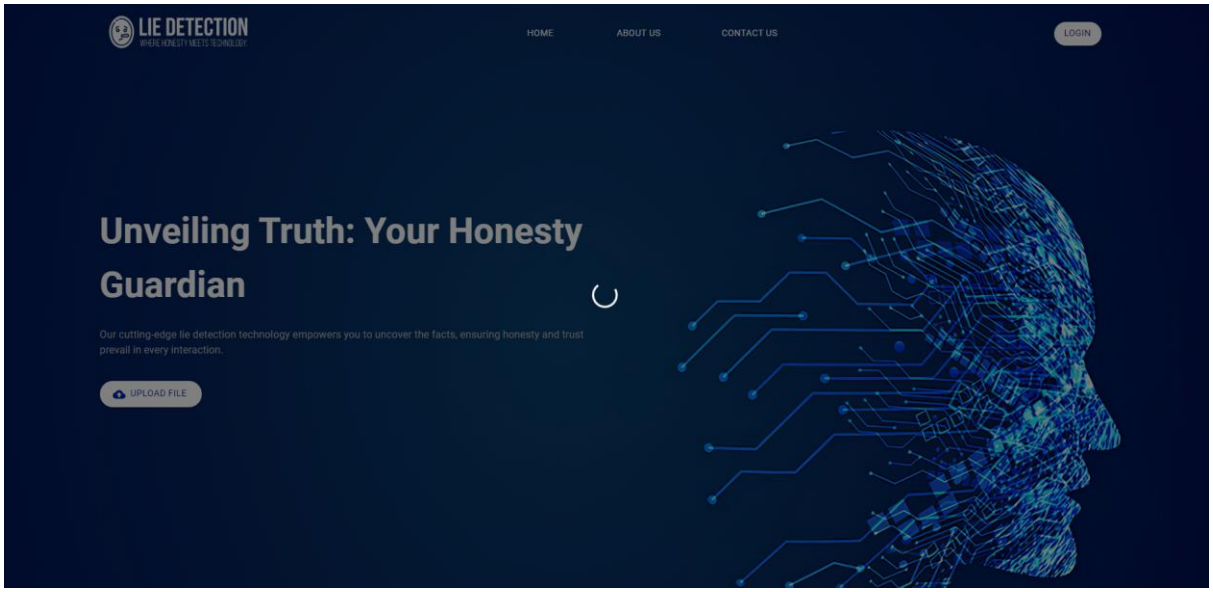

Figure 7.3: Loading screen

4. The user can see the result of the video classification, the user also has the option to get more details about the prediction.

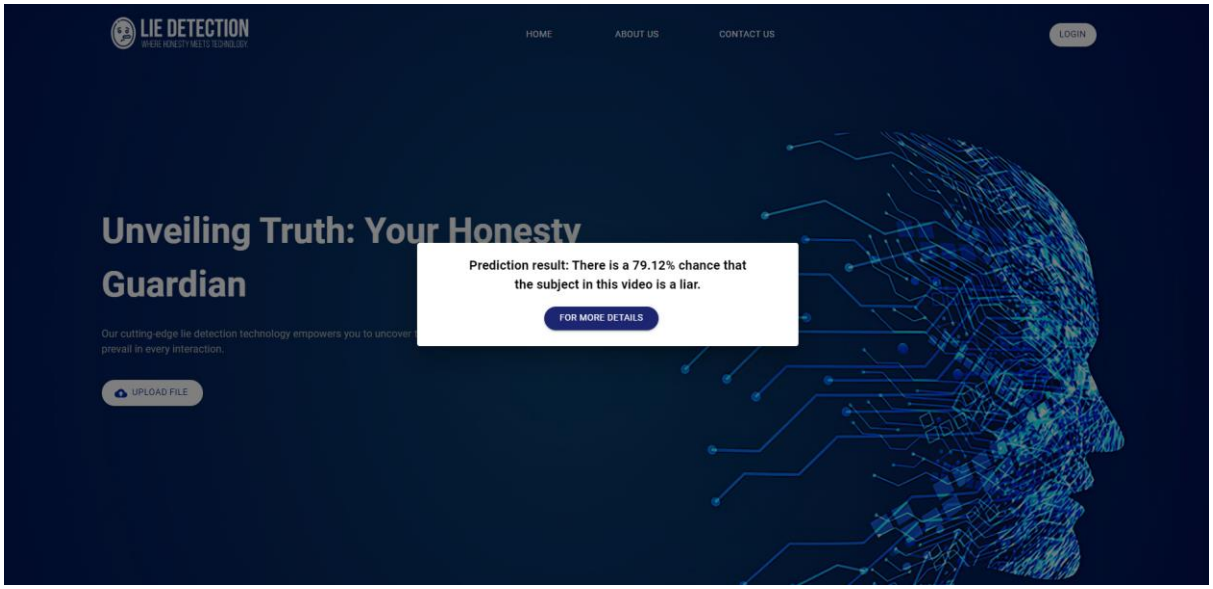

Figure 7.4: Results screen

5. If the user clicks “for more details”, they can see detailed results of each individual modality.

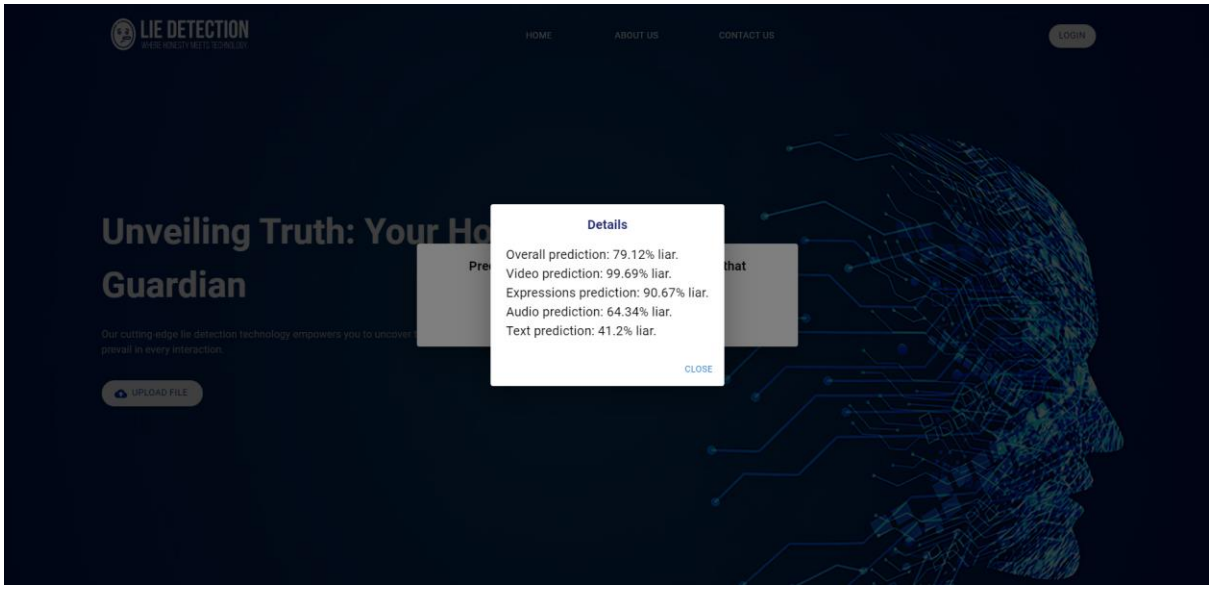

Figure 7.5: Detailed results screen

8. Conclusion & Future Work

8.1 Conclusion

The aim of this project was to explore the limits that can be reached in the domain of deception detection using state of the art deep learning and machine learning methodologies, our work was inspired by the works recently published achieving accuracies above 80% such as those in [5], [9], [10], [11], [20].

Multimodal fusion was the major premise behind the study; thus, we first started on singular modalities achieving an accuracy of **87.5%** on video frames modality using 3D CNNs, **79.1%** on Audio data using bidirectional LSTMS, **79.1%** on Text data, and **75%** on micro expressions that we extracted. After our efforts on individual modalities, we considered multimodal fusion, particularly late fusion, and our best model was implemented using majority voting on visual, vocal and linguistic features achieving an accuracy of **95.8%**.

8.2 Future work

1. Explore other data fusion techniques such as early fusion and hybrid fusion.

2. Benchmark the proposed architectures on larger datasets that are not strictly courtroom themed.

3. Gather a large high quality deception detection dataset as there is a lack of high quality data when it comes to the domain of deception detection.

4. Try more complex neural network architectures and different machine learning classifiers.

5. Experiment with making a real-time deception detection system.

6. Explore the possibility of extracting heart rate signals from videos using remote photoplethysmography [15], [16], [17], [18], and exploring the relationship between heart rate signals and deception.

References

[1] Bond Jr, C.F., DePaulo, B.M.: Accuracy of deception judgments. Personality and Social Psychology Review 10 (2006) 214–234.

[2] DePaulo, B.M., Lindsay, J.J., Malone, B.E., Muhlenbruck, L., Charlton, K. and Cooper, H., 2003. Cues to deception. *Psychological bulletin*, *129*(1), p.74.

[3] Perez-Rosas, V., Abouelenien, M., Mihalcea, R., Burzo, M.: Deception detection using real-life trial data. In: Proceedings of the 2015 ACM on International Conference on Multimodal Interaction, ACM (2015).

[4] Pérez-Rosas, V., Abouelenien, M., Mihalcea, R., Xiao, Y., Linton, C.J. and Burzo, M., 2015, September. Verbal and nonverbal clues for real-life deception detection. In Proceedings of the 2015 conference on empirical methods in natural language processing (pp. 2336-2346).

[5] Şen, M.U., Perez-Rosas, V., Yanikoglu, B., Abouelenien, M., Burzo, M. and Mihalcea, R., 2020. Multimodal deception detection using real-life trial data. *IEEE Transactions on Affective Computing*, *13*(1), pp.306-319.

[6] A. Vrij. Detecting Lies and Deceit: The Psychology of Lying and the Implications for Professional Practice. Wiley series in the psychology of crime, policing and law. Wiley, 2001.

[7] T. Gannon, A. Beech, and T. Ward. Risk Assessment and the Polygraph, pages 129–154. John Wiley and Sons Ltd, 2009

[8] Bareeda, E.F., Mohan, B.S. and Muneer, K.A., 2021, May. Lie detection using speech processing techniques. In Journal of Physics: Conference Series (Vol. 1921, No. 1, p. 012028). IOP Publishing.

[9] Ahmed, H.U.D., Bajwa, U.I., Zhang, F. and Anwar, M.W., 2021. Deception detection in videos using the facial action coding system. arXiv preprint arXiv:2105.13659.

[10] Mathur, L. and Matarić, M.J., 2020, October. Introducing representations of facial affect in automated multimodal deception detection. In Proceedings of the 2020 International Conference on Multimodal Interaction (pp. 305-314).

[11] Krishnamurthy, G., Majumder, N., Poria, S. and Cambria, E., 2018, March. A deep learning approach for multimodal deception detection. In International Conference on Computational Linguistics and Intelligent Text Processing (pp. 87-96). Cham: Springer Nature Switzerland.

[12] Rill-García, R., Jair Escalante, H., Villasenor-Pineda, L. and Reyes-Meza, V., 2019. High-level features for multimodal deception detection in videos. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (pp. 0-0).

[13] S. Feng, R. Banerjee, and Y. Choi. Syntactic stylometry for deception detection. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2, ACL '12, pages 171–175, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[14] Barathi, C.S., 2016. Lie detection based on facial micro expression body language and speech analysis. International Journal of Engineering Research & Technology, 5(2), pp.337- 343.

[15] Ni, A., Azarang, A. and Kehtarnavaz, N., 2021. A review of deep learning-based contactless heart rate measurement methods. Sensors, 21(11), p.3719.

[16] Martinez-Delgado, G.H., Correa-Balan, A.J., May-Chan, J.A., Parra-Elizondo, C.E., GuzmanRangel, L.A. and Martinez-Torteya, A., 2022. Measuring heart rate variability using facial video. Sensors, 22(13), p.4690.

[17] Yu, Z., Peng, W., Li, X., Hong, X. and Zhao, G., 2019. Remote heart rate measurement from highly compressed facial videos: an end-to-end deep learning solution with video enhancement. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 151-160).

[18] Zhan, Q., Wang, W. and de Haan, G., 2020. Analysis of CNN-based remote-PPG to understand limitations and sensitivities. Biomedical optics express, 11(3), pp.1268-1283.

[19] A. Khaled, G. Walaa, M. Medhat, R. Alaa, and T. Ehab, "Lie Detection System," Ain Shams University, Faculty of Computer & Information Sciences, Scientific Computing Department.

[20] T. Yang, G.-M. Liu, and S. C.-H. . Huang, “Multimodal Deception Detection in Videos via Analyzing Emotional State-based Feature,” arXiv:2104.08373 [cs], Apr. 2021.

[21] H. Karimi, J. Tang, and Y. Li, “Toward End-to-End Deception Detection in Videos,” IEEE Xplore, Dec. 01, 2018.

[22] “Py-FEAT: Python Facial Expression Analysis Toolbox,” GitHub, Feb. 23, 2022. license - cosanlab/py-feat · GitHub.

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

[24] Goodfellow, I., Bengio, Y., and Courville, A., 2017. Deep Learning. MIT Press.

[25] Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Processing Letters, 23(10), 1499-1503.

[26] Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001) (Vol. 1, pp. I-I). IEEE.

[27] Abdul, Z.K. and Al-Talabani, A.K., 2022. Mel frequency cepstral coefficient and its applications: A review. IEEE Access, 10, pp.122136-122158.

[28] Prince, E.B., Martin, K.B., Messinger, D.S. and Allen, M., 2015. Facial action coding system. Environmental Psychology & Nonverbal Behavior, 1.

[29] Pawłowski, M., Wróblewska, A. and Sysko-Romańczuk, S., 2023. Effective techniques for multimodal data fusion: A comparative analysis. Sensors, 23(5), p.2381.

[30] Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press

[31] Cristianini, N., & Shawe-Taylor, J. (2000). An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press

[32] Quinlan, J. R. (1986). Induction of decision trees. Machine Learning, 1(1), 81-106.

[33] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5-32.

[34] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794).

[35] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010* (pp. 177-186). Physica-Verlag HD.

[36] Jurafsky, D., & Martin, J. H. (2020). Speech and Language Processing (3rd ed.). Prentice Hall.

[37] Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics

ملخص

الكذب شائع في حياتنا اليومية، وغالبًا ما يكون له عواقب خطيرة في مجالات حيوية مثل تحقيقات الشرطة وأمن المطارات والمحاكم. قدرة الإنسان على كشف الكذب بدون أدوات خاصة محدودة للغاية.

السلوك الكاذب يتسبب في إشارات متعدده عند الاشخاص، منها الإشارات اللفظية والظاهرية. على سبيل المثال، قد يظهر الأفراد المخادعون تناقضات في اختياراتهم اللغوية، تغييرات في نبرة الصوت او بعض التردد. من الناحية الظاهرية، قد تظهرعليهم العصبية الزائدة ،او يتجنبون الاتصال البصري، أو يظهرون تعابير وجه غير متسقة مع كلماتهم. اكتشاف هذه الإشارات معقد ويتطلب عادة تدريبًا مهنيًا، وذلك قد يكون مكلفًا ويستغرق الكثير من الوقت، وحتى مع ذلك، فإن اداء ذلك بدقة غير مؤكد.

كان هدفنا هو تطوير نظام يعتمد على الذكاء الاصطناعي قادر علي استغلال الإشارات البصرية واللفظية لكشف الكذب. تهدف هذه الحلول إلى أن تكون أكثر فعالية من حيث انها افل تكلفة من توظيف المختصين, الي جانب كونها مفيدة في مجالات مختلفة من ضمنها تحقيقات الشرطة.

في بحثنا، استخدمنا اسلوبا يعتمد علي البيانات متعددة النوع Multimodal، يدمج بيانات الفيديو، إشارات الصوت، الخصائص اللغوية من النص، و التعبيرات الدقيقة Micro expressions لتعزيز كشف الكذب. درسنا الابحاث السابقة وحاولنا تحسينها. عملنا من الصفر لبناء نماذج مماثلة واختبرناها بأنفسنا لمعرفة تأثير البيانات المتعددة النوع Multimodal features. حقق نظامنا دقة بلغت 95.8٪ على قاعدة بيانات Real Life Detection Dataset Deception باستخدام الدمج المتأخر Late Fusion للإشارات البصرية، الصوتية واللغوية.