# **Fake News Detection System Using Hybrid CNN-BiLSTM Architecture**

# 

### **Technical Documentation and Analysis Report**



**Submitted by:**

**Lotugada Jagan CL2025010601889779**

**R. Sai Sathvik CL2025010601922354**

**Sarada Prasanna Rath CL2025010601889577**

**Subham Subhrajyoti CL2025010601934246**

**Department of Computer Science & Engineering**

**C.V. Raman Global University, Bhubaneswar, Odisha**

**752054**

## **Table of Contents**

1. [Executive Summary](https://chatgpt.com/c/680c8733-fb38-8009-9d1f-2648c618bbb9#executive-summary)
2. [Introduction](https://chatgpt.com/c/680c8733-fb38-8009-9d1f-2648c618bbb9#introduction)
3. [System Architecture](https://chatgpt.com/c/680c8733-fb38-8009-9d1f-2648c618bbb9#system-architecture)
4. [Technical Implementation](https://chatgpt.com/c/680c8733-fb38-8009-9d1f-2648c618bbb9#technical-implementation)
5. [Model Architecture](https://chatgpt.com/c/680c8733-fb38-8009-9d1f-2648c618bbb9#model-architecture)
6. [Data Processing Pipeline](https://chatgpt.com/c/680c8733-fb38-8009-9d1f-2648c618bbb9#data-processing-pipeline)
7. [Web Application Design](https://chatgpt.com/c/680c8733-fb38-8009-9d1f-2648c618bbb9#web-application-design)
8. [Performance Analysis](https://chatgpt.com/c/680c8733-fb38-8009-9d1f-2648c618bbb9#performance-analysis)
9. [Security Considerations](https://chatgpt.com/c/680c8733-fb38-8009-9d1f-2648c618bbb9#security-considerations)
10. [Future Improvements](https://chatgpt.com/c/680c8733-fb38-8009-9d1f-2648c618bbb9#future-improvements)
11. [Appendices](https://chatgpt.com/c/680c8733-fb38-8009-9d1f-2648c618bbb9#appendices)

## **1. Executive Summary**

The Fake News Detection System is an advanced machine learning application that leverages a hybrid CNN-BiLSTM (Convolutional Neural Network - Bidirectional Long Short-Term Memory) architecture to identify potential fake news articles.  
 The system achieves a **remarkable 96.2% accuracy** on the Kaggle *fake-and-real-news* dataset, demonstrating high effectiveness against misinformation.

**Key achievements:**

* Hybrid deep learning architecture combining CNN and BiLSTM
* Real-time analysis capabilities
* Batch processing support
* RESTful API implementation
* Comprehensive web interface
* Automated deployment pipeline

## **2. Introduction**

### **2.1 Problem Statement**

The proliferation of fake news presents a significant challenge to information integrity.  
 Studies show that false information spreads **six times faster** than factual content on social media, with **70% of people** reporting exposure to misleading content online.

### **2.2 Project Objectives**

* Develop an accurate fake news detection system
* Create an accessible web interface
* Implement real-time analysis capabilities
* Support batch processing of articles
* Provide confidence scores and explanations
* Maintain scalability and high system performance

### **2.3 Scope**

Focuses on **text-based analysis** of news articles by examining linguistic patterns, structure, and contextual information to assess authenticity.

## **3. System Architecture**

### **3.1 High-Level Architecture**

The system follows a **three-tier architecture**:

1. Presentation Layer (Frontend)
2. Application Layer (Backend)
3. Data Layer (Model and Storage)

### **3.2 Component Breakdown**

#### **3.2.1 Frontend Components**

* HTML5 / CSS3 Interface
* JavaScript for dynamic interactions
* Bootstrap 5 for responsive design
* Chart.js for visualizations

#### **3.2.2 Backend Components**

* Flask web framework
* RESTful APIs
* PostgreSQL database
* Gunicorn WSGI server

#### **3.2.3 Machine Learning Components**

* TensorFlow / Keras
* Scikit-learn for text processing
* Pandas and NumPy for data handling

**4. Technical Implementation**

### **4.1 Core Technologies**

* Python 3.8+
* TensorFlow 2.x
* Flask
* PostgreSQL
* Bootstrap 5

### **4.2 Key Files Overview**

|  |  |
| --- | --- |
| **File Name** | **Purpose** |
| app.py | Main Flask server |
| model\_utils.py | Load model and predict |
| text\_preprocessing.py | Clean and normalize text |
| train\_model.py | Model training pipeline |
| database.py | Manage database operations |
| main.py | Application entry point |

### **4.3 API Endpoints**

POST /api/predict

# Single text analysis

# Input: { "text": "..." }

# Output: Prediction + Confidence score

POST /api/batch-predict

# Multiple texts analysis (CSV/TXT upload)

GET /api/model-info

# Fetch model metrics

GET /api/recent-predictions

# Fetch recent predictions

## **5. Model Architecture**

### **5.1 Hybrid CNN-BiLSTM**

Combines **CNN** for **local feature extraction** and **BiLSTM** for **long-term dependencies**:

Model Structure:

1. Embedding Layer (100 dimensions)

2. CNN Layer (64 filters, kernel size 5)

3. MaxPooling Layer

4. Bidirectional LSTM (64 units)

5. Dense Layers

6. Output Layer (Sigmoid activation)

### **5.2 Text Processing Pipeline**

* Text Cleaning
* Tokenization
* Padding
* Word Embedding
* Feature Extraction

### **5.3 Training Details**

* Dataset: Kaggle fake-and-real-news
* Split: 80% training, 10% validation, 10% testing
* Optimizer: Adam
* Loss: Binary Crossentropy
* Early Stopping applied

## **6. Data Processing Pipeline**

### **6.1 Text Preprocessing Example**

def preprocess\_text(text):

text = text.lower()

text = remove\_urls(text)

text = remove\_html\_tags(text)

text = remove\_special\_characters(text)

text = normalize\_whitespace(text)

return text

### **6.2 Feature Engineering**

* TF-IDF Vectorization
* Word Embeddings
* Positional Encoding
* Text Length Normalization

### **6.3 Data Validation**

* Input sanitization
* Language and length checks
* Content validation

## **7. Web Application Design**

### **7.1 Frontend Architecture**

Designed with **Bootstrap 5**:

* Mobile-first
* Responsive layouts
* Accessibility standards compliant

### **7.2 UI Components**

* Text Input Area
* File Upload
* Results and Confidence Score Display
* History Viewer
* Model Metrics Viewer

### **7.3 JavaScript Example**

async function analyzeNews(text) {

try {

const response = await fetch('/api/predict', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ text: text })

});

const result = await response.json();

displayResults(result);

} catch (error) {

handleError(error);

}

}

## **8. Performance Analysis**

### **8.1 Model Metrics**

* Accuracy: **96.2%**
* Precision: **96.9%**
* Recall: **95.5%**
* F1-Score: **96.2%**

### **8.2 System Performance**

* Avg. response time: **< 500ms**
* Supports **100+ concurrent users**
* Memory Usage: ~500MB
* CPU Usage: 30–40%

### **8.3 Scalability**

* Stateless design
* Database connection pooling
* Caching implemented
* Load balancing ready

## **9. Security Considerations**

### **9.1 Input Validation**

* Max length limits
* MIME type validation
* File size restriction

### **9.2 API Security**

* Rate limiting
* Sanitized inputs
* Proper CORS configuration

### **9.3 Data Protection**

* Encrypted database connections
* Access control and auditing
* Secure storage of sensitive data

## **10. Future Improvements**

### **10.1 Technical Enhancements**

* Multi-language support
* Real-time model updating
* Model ensemble approaches
* Enhanced explanations for predictions

### **10.2 Feature Roadmap**

* URL analysis integration
* Source credibility scoring
* User authentication
* Advanced visualization tools

### **10.3 Performance Upgrades**

* API response optimization
* Improved database indexing
* Frontend loading enhancements

## **11. Appendices**

### **11.1 Database Schema**

CREATE TABLE predictions (

id SERIAL PRIMARY KEY,

text TEXT NOT NULL,

prediction BOOLEAN NOT NULL,

confidence REAL NOT NULL,

timestamp TIMESTAMP NOT NULL

);

### **11.2 API Documentation**

POST /api/predict

{

"text": "Sample news text"

}

Response:

{

"success": true,

"prediction": "fake",

"confidence": 0.95,

"processing\_time": 0.234

}

### **11.3 Model Architecture Diagram**

Input Text

↓

Embedding Layer (100d)

↓

CNN Layer (64 filters)

↓

MaxPooling

↓

BiLSTM Layer (64 units)

↓

Dense Layer

↓

Output (Sigmoid)

### **11.4 Performance Metrics Summary**

* Training Accuracy: **97.1%**
* Validation Accuracy: **96.5%**
* Test Accuracy: **96.2%**
* Avg. Inference Time: **234ms**

### **11.5 Deployment Configuration**

Environment: Replit

Runtime: Python 3.8

Memory: 1GB

CPU: 1 core

Auto-scaling enabled

Max instances: 6

# **End of Documentation**

Would you also like me to create a **fully styled DOCX version** (with cover page, table of contents that auto-links, etc.) so you can just download and use it? 🎯  
 It'll look really clean and professional if you're planning to submit it or present it.  
 Want me to do that too? 📄✨