Deep Learning-Based Detection of Motor Biomarkers for Autism from Children's Video Recordings

Overview

This implementation presents a hybrid deep learning approach for detecting autism spectrum disorder (ASD) in children through analysis of motor biomarkers extracted from video recordings. The system combines BiLSTM, CNN, and Multi-Head Attention mechanisms in an ensemble architecture to achieve high-accuracy autism detection.

System Requirements

Hardware Specifications

- **Processor**: Intel(R) Core(TM) i5-120U @ 1.40 GHz
- **Memory**: 16.0 GB RAM (15.6 GB usable)
- **Storage**: 423 GB available space
- **Graphics**: Intel(R) Graphics (integrated)
- **System Type**: 64-bit operating system, x64-based processor

Software Environment

- **Operating System**: Windows 11 (64-bit)
- **Python Version**: 3.8+ (recommended 3.9)
- **Platform**: Windows x64

Dependencies and Libraries

Core Deep Learning Framework

Plain Text

```
tensorflow>=2.8.0
keras>=2.8.0
```

Data Processing and Analysis

Plain Text

```
numpy>=1.21.0
pandas>=1.3.0
scikit-learn>=1.0.0
scipy>=1.7.0
```

Computer Vision and Pose Detection

Plain Text

```
mediapipe>=0.8.9
opencv-python>=4.5.0
```

Data Balancing

Plain Text

```
imbalanced-learn>=0.8.0
```

Visualization

Plain Text

```
matplotlib>=3.5.0
seaborn>=0.11.0
```

Utilities

Plain Text

```
pickle
os
```

Installation

Install dependencies

Bash

```
pip install tensorflow keras numpy pandas scikit-learn scipy mediapipe
opencv-python imbalanced-learn matplotlib seaborn
```

Project Structure

Plain Text

```
├── feature_engineering.py           # Behavioral biomarker extraction
├── train_hybrid_model_fixed2.py     # Main hybrid model training
├── evaluate_ensemble_model.py       # Ensemble model evaluation
├── evaluate_models.py               # Individual model evaluation
├── preprocess_data.py               # Data preprocessing pipeline
├── create_pose_landmark_visualization.py   # Visualization tools
├── kare_bazli_ozellik_cikarimi.py   # Frame-based feature extraction
├── models/                          # Trained model files
└── results/                         # Evaluation results
```

Usage

1. Data Preprocessing

Bash

```
python preprocess_data.py
```

2. Feature Extraction

```bash
python feature_engineering.py
```

3. Model Training

```bash
python train_hybrid_model_fixed2.py
```

4. Model Evaluation

```bash
python evaluate_ensemble_model.py
python evaluate_models.py
```

5. Visualization

```bash
python create_pose_landmark_visualization.py
```

Model Architecture

Hybrid Ensemble Components

1. **BiLSTM + CNN + Attention Model (50% weight)**
   - Bidirectional LSTM layers (64, 32 units)
   - 1D CNN with multiple kernel sizes (3, 5, 7)
   - Multi-head attention mechanism (4 heads)
2. **GRU Model (30% weight)**

- Two GRU layers (64, 32 units)
- Dropout regularization

3. **CNN Model (20% weight)**
   - Three 1D convolutional layers (64, 128, 128 filters)
   - Global pooling layers

Training Configuration

- **Optimizer**: Adam (β_1=0.9, β_2=0.999)
- **Learning Rate**: 0.001
- **Batch Size**: 32
- **Max Epochs**: 50
- **Loss Function**: Binary Cross-entropy
- **Regularization**: Dropout (20-40%)
- **Callbacks**: EarlyStopping, ReduceLROnPlateau

Performance Metrics

Ensemble Model Results

- **Accuracy**: 97.11%
- **Precision**: 98.31%
- **Recall**: 95.87%
- **F1-Score**: 97.07%
- **ROC-AUC**: 98.24%
- **Specificity**: 98.35%

Individual Model Performance

Model	Accuracy	Precision	Recall	F1-Score	AUC
BiLSTM+CNN+Attention	95.0%	96.0%	94.0%	95.0%	97.0%
GRU	92.0%	93.0%	91.0%	92.0%	95.0%
CNN	89.0%	90.0%	88.0%	89.0%	93.0%

Behavioral Biomarkers

Feature Categories

1. **Movement Parameters**: Velocity, acceleration, jerk
2. **Statistical Features**: Mean, std, skewness, kurtosis
3. **Frequency Domain**: Dominant frequency, spectral centroid, bandwidth
4. **Coordination Indices**: Left-right correlation, synchronization score
5. **Repetitiveness Measures**: Repetition count, regularity index

Data Requirements

Input Format

- **Video Files**: MP4, AVI formats supported
- **Frame Rate**: 10 FPS (recommended)
- **Duration**: 30 seconds minimum
- **Content**: Upper body movements of children (1-10 years)

Dataset Structure

- **Training Set**: 80% (1,934 sequences)

- **Test Set**: 20% (484 sequences)
- **Class Balance**: SMOTE applied for balanced training

Reproducibility

Pseudocode Algorithms

Complete pseudocode algorithms are provided in Appendix A of the accompanying research paper, covering:

- Feature extraction pipeline
- Hybrid model architecture
- Ensemble learning strategy
- Cross-validation procedures
- Statistical analysis methods

External Validation on Real-World Videos

We evaluated the trained ensemble on an out-of-distribution set of 42 public videos (19 ASD, 23 TD). Four additional videos were excluded due to codec/format issues.

On this real-world set, the ensemble achieved:

- Balanced Accuracy: 83.4%
- Overall Accuracy: 83.3%
- Sensitivity (ASD): 84.2%
- Specificity (TD): 82.6%
- Precision (PPV): 80.0%
- Negative Predictive Value (NPV): 86.4%
- F1-Score: 82.1%

(See Table 5 in the manuscript for the full summary.)
Test video links are listed in test_autism_urls.txt and test_normal_urls.txt.

Notes. The real-world evaluation involves diverse recording conditions and viewpoints; as expected, performance is lower than in-distribution results. Pose landmarks focus on the upper body, which may omit other cues (e.g., facial/eye movements, lower-limb dynamics).

Post-training Ensemble Weight Optimization (Grid Search)

After training the three base learners (Hybrid BiLSTM+CNN+Attention, GRU, CNN), we tuned the linear ensemble weights on the held-out validation split using a simple grid search under the constraint w1+w2+w3=1w_1 + w_2 + w_3 = 1w1 +w2+w3=1.
The best configuration used for reporting on the external set is:

- CNN: 0.753
- BiLSTM+CNN+Attention: 0.120
- GRU: 0.127
- Decision threshold: 0.550 (selected from the validation PR curve)

The final prediction is computed as:

$p_{ens} = 0.753 * p_{CNN} + 0.120 * p_{Hybrid} + 0.127 * p_{GRU}$

$\hat{y} = 1[\, p_{ens} \geq 0.550 \,]$

(Weights and threshold are summarized in Table 5.)

Reproducibility.
Use the released checkpoints in models.zip and follow the same preprocessing.
To re-run the external test, load the three model probabilities, sweep weights on a simplex (e.g., step 0.01) on the validation split, select the triple that maximizes balanced accuracy, fix the decision threshold at 0.550, then report metrics on the held-out real-world set.

School Screening with Autism Detection Model

This section provides a practical guide for using the deep learning-based autism detection model for school screening purposes. The system is designed to be user-friendly and includes automated processes for video analysis and report generation.

STEP 1: Python Libraries

Ensure you have the necessary Python libraries installed. You can install them using pip:

```bash
pip install tensorflow opencv-python mediapipe numpy pathlib pandas openpyxl
```

To verify the installation, run the following Python code:

```python
import tensorflow as tf
import cv2
import mediapipe as mp
import numpy as np
import pandas as pd
from pathlib import Path
print("All libraries are ready!")
```

STEP 2: Prepare Files

First, create a working directory and navigate into it:

```bash
mkdir school_screening
cd school_screening
```

Then, copy the `real_ensemble_complete_system.zip` file into this directory and unzip it:

```bash
unzip real_ensemble_complete_system.zip
```

This will extract the following files:

```
school_screening/
 ├── simple_real_ensemble_model.h5 (227 KB - not used)
 ├── simple_real_ensemble_config.json (1.6 KB - IMPORTANT!)
 ├── simple_real_ensemble_video_system.py (21 KB - MAIN SYSTEM)
 └── simple_real_easy_video_tester.py (7.8 KB - TEST TOOL)
```

STEP 3: Prepare Individual Models

The system requires individual model files (`cnn_model.h5`, `bilstm_cnn_attention_model.h5`, `gru_model.h5`). If these are missing, you need to update the `simple_real_ensemble_config.json` file to point to their correct paths. Copy these three model files into the same directory as the unzipped system files.

Edit `simple_real_ensemble_config.json`:

```json
{
  "individual_model_paths": {
    "cnn_model": "./cnn_model.h5",
    "bilstm_cnn_attention_model": "./bilstm_model.h5",
    "gru_model": "./gru_model.h5"
  }
}
```

STEP 4: Prepare Videos

Video Requirements:
*   **Format:** MP4, AVI, MOV
*   **Duration:** 15-30 seconds
*   **Quality:** 720p or higher
*   **Content:** Child's full body visible
*   **Movement:** Natural play, walking

Video Shooting Guidelines:
*   **Correct:** Child standing, playing; stable camera (2-3 meters distance); good lighting; full body in frame; 20-30 seconds duration.
*   **Incorrect:** Only face visible; child sitting/motionless; dark environment; very short video (<10 seconds).

STEP 5: Single Video Test

From Command Line (Easiest):

```bash
python simple_real_easy_video_tester.py ahmet.mp4
```

With Python Code:

```python
from simple_real_ensemble_video_system import
SimpleRealEnsembleVideoSystem

# Initialize the system
system = SimpleRealEnsembleVideoSystem()

# Test video
result = system.predict_video("ahmet.mp4")

if result['success']:
    print(f"Student: Ahmet")
    print(f"Result: {result['prediction']['label']}")
    print(f"Probability: {result['prediction']['probability']:.3f}")
    print(f"Confidence: {result['prediction']['confidence']:.3f}")
    if result['prediction']['probability'] > 0.550:
        print("AUTISM RISK - Expert evaluation recommended")
    else:
        print(" NORMAL DEVELOPMENT")
else:
    print(f" Error: {result['error']}")

# Close the system
system.close()
```

STEP 6: Class Screening

Prepare Video Folder Structure:

```
school_videos/
 ├── 1A_class/
 │    ├── ahmet.mp4
 │    ├── ayse.mp4
 │    ├── mehmet.mp4
 │    └── fatma.mp4
 ├── 1B_class/
 │    ├── ali.mp4
```

```
│   └── zeynep.mp4
└── results/
```

Batch Test Script:

```python
import os
import json
from pathlib import Path
from simple_real_ensemble_video_system import
SimpleRealEnsembleVideoSystem

def class_screening(video_folder):
    """Test all class videos"""
    system = SimpleRealEnsembleVideoSystem()
    video_files = []
    for ext in ['.mp4', '.avi', '.mov']:
        video_files.extend(Path(video_folder).glob(f"**/*{ext}"))

    print(f" {len(video_files)} videos found")
    results = []
    for i, video_path in enumerate(video_files, 1):
        print(f"\n[{i}/{len(video_files)}] Test: {video_path.name}")
        result = system.predict_video(str(video_path))
        if result['success']:
            screening_result = {
                'video_name': video_path.name,
                'student_name': video_path.stem,
                'class': video_path.parent.name,
                'result': result['prediction']['label'],
                'probability': round(result['prediction']['probability'], 3),
                'confidence': round(result['prediction']['confidence'], 3),
                'autism_risk': result['prediction']['probability'] > 0.550
            }
            results.append(screening_result)
            risk_status = "⚠ RISK" if screening_result['autism_risk'] else "NORMAL"
            print(f" {screening_result['student_name']}: {screening_result['result']}
({screening_result['probability']}) {risk_status}")
        else:
            print(f" Error: {result['error']}")

    with open('screening_results.json', 'w', encoding='utf-8') as f:
        json.dump(results, f, indent=2, ensure_ascii=False)
    system.close()

    total = len(results)
```

```
    risk_count = len([s for s in results if s['autism_risk']])
    print(f"\n SCREENING RESULTS SUMMARY:")
    print(f" Total students: {total}")
    print(f" Autism risk: {risk_count} ({risk_count/total*100:.1f}%)")
    print(f" Normal: {total-risk_count} ({(total-risk_count)/total*100:.1f}%)")
    return results

# Usage example:
# results = class_screening("school_videos")
```

STEP 7: Excel Report

Generate Excel Report:

```python
import pandas as pd

def generate_excel_report(results):
    """Export results to Excel"""
    df = pd.DataFrame(results)
    df['Status'] = df['autism_risk'].apply(lambda x: 'RISK' if x else 'NORMAL')

    with pd.ExcelWriter('school_screening_report.xlsx', engine='openpyxl') as
writer:
        df[['student_name', 'class', 'result', 'probability', 'Status']].to_excel(
            writer, sheet_name='Screening_Results', index=False
        )

        class_summary = df.groupby('class').agg({
            'student_name': 'count',
            'autism_risk': 'sum'
        }).rename(columns={
            'student_name': 'Total_Students',
            'autism_risk': 'Risk_Count'
        })
        class_summary['Risk_Rate'] = (class_summary['Risk_Count'] /
class_summary['Total_Students'] * 100).round(1)
        class_summary.to_excel(writer, sheet_name='Class_Summary')

        overall_summary = pd.DataFrame({
            'Metric': ['Total Students', 'Autism Risk', 'Normal', 'Risk Rate (%)'],
            'Value': [
                len(df),
                df['autism_risk'].sum(),
                len(df) - df['autism_risk'].sum(),
                f"{df['autism_risk'].sum()/len(df)*100:.1f}%"
```

```
            ]
        })
        overall_summary.to_excel(writer, sheet_name='Overall_Summary',
index=False)

    print("Excel report created: school_screening_report.xlsx")

# Usage example:
# generate_excel_report(results)
```

STEP 8: Interpret Results

Result Interpretation:
*   ** AUTISM RISK (Probability > 0.550):** Schedule a meeting with the family, recommend expert evaluation, inform about early intervention, re-evaluate after 3-6 months.
*   ** NORMAL (Probability < 0.550):** Routine follow-up, developmental monitoring, family information.

Points to Note:
*   There is a 15.8% false negative risk.
*   This is a **SCREENING** tool, not a **DIAGNOSIS**.
*   Expert approval is required.
*   Regular updates are necessary.

STEP 9: Troubleshooting

Common Errors:
*   `"ModuleNotFoundError: No module named 'tensorflow'"`
    *   **Solution:** `pip install tensorflow`
*   `"FileNotFoundError: cnn_model.h5"`
    *   **Solution:** Add individual model files.
*   `"Video could not be loaded"`
    *   **Solution:** Check video format (MP4 recommended).
*   `"Pose detection failed"`
    *   **Solution:** Improve video quality, adjust lighting.

Summary Checklist

*   Python libraries installed
*   Zip file unzipped
*   Individual model files added
*   Config file updated
*   Tested with a single video
*   Batch test script prepared
*   Excel report system set up

* Result interpretation learned
* Expert communication network established
* Ethical rules defined

SYSTEM READY!
You can now perform school screenings!

Contact

For questions or issues, please contact:

- Email: yelda.firat@mudanya.edu.tr