Video Analysis AI Enhancement - Implementation Guide

Overview

This document outlines the comprehensive Video Analysis AI Enhancement system implemented for the Mindful Champion pickleball coaching app. The system uses real TensorFlow.js pose detection to analyze player technique and provide personalized feedback.

Features Implemented

1. TensorFlow.js Pose Detection ✅

Location: `lib/video-analysis/pose-detection/tensorflow-pose-detector.ts`

- **Real pose detection** using TensorFlow.js MoveNet model
- Processes video frames to detect player body positions
- Extracts keypoints for 17 body parts (nose, eyes, shoulders, elbows, wrists, hips, knees, ankles)
- Confidence scoring for each detected pose
- Frame sampling for performance optimization

Key Methods:
- `initialize()` - Loads the TensorFlow model
- `detectPosesInVideo(videoPath)` - Analyzes full video and returns pose data
- `dispose()` - Cleans up resources

2. Pickleball-Specific Technique Analysis ✅

Location: `lib/video-analysis/pose-detection/pickleball-technique-analyzer.ts`

Analyzes detected poses to extract pickleball-specific metrics:

Serve Mechanics

- Arm angle during serve
- Follow-through quality
- Body rotation

Footwork Patterns

- Stance width and balance
- Split-step timing
- Agility and movement quality

Paddle Position

- Paddle height
- Paddle angle consistency
- Ready position maintenance

Body Positioning

- Body alignment

- Center of gravity
- Overall positioning

Key Methods:
- `analyzePoseSequence(poses)` - Analyzes sequence of poses for technique metrics
- `detectShot(prevPose, currentPose)` - Detects when shots occur
- `classifyShotType(pose)` - Classifies shot as serve, forehand, backhand, volley, dink, smash, or lob

3. Visual Overlays ✅

Location: `lib/video-analysis/visual-overlays/pose-overlay-generator.ts`

Creates visual overlays showing:
- **Skeleton tracking** - Green lines connecting joints
- **Keypoint markers** - Red dots at key body positions
- **Angle measurements** - Displays elbow/arm angles in degrees
- **Form indicators** - Color-coded circles (green = good form, yellow = okay, red = needs work)

Component: `components/video-analysis/enhanced/pose-overlay-viewer.tsx`

Interactive video player with:
- Toggle overlays on/off
- Play/pause controls
- Timeline scrubbing
- Overlay legend

4. Personalized Drill Recommendations ✅

Location: `lib/video-analysis/drills/drill-recommendation-engine.ts`

- Library of 10+ pickleball-specific drills
- Categorized by: serve, footwork, volleys, dinks, groundstrokes, positioning
- Difficulty levels: beginner, intermediate, advanced
- Each drill includes:
- Detailed instructions (4-5 steps)
- Focus areas
- Expected improvement metrics
- Duration and priority

Component: `components/video-analysis/enhanced/drill-recommendations.tsx`

Two-tab interface:
1. **Top Drills** - Personalized recommendations based on analysis
2. **Weekly Plan** - 4-day training schedule with distributed drills

5. Comparison Metrics ✅

Location: `lib/video-analysis/comparison/metrics-comparison.ts`

Benchmark Comparison

Compares user performance against skill-level benchmarks:
- Beginner (60-65 avg scores)
- Intermediate (73-77 avg scores)
- Advanced (83-87 avg scores)
- Professional (91-94 avg scores)

Provides:
- Per-metric comparison (13 metrics)
- Overall percentile ranking
- Identification of strengths and weaknesses

Component: `components/video-analysis/enhanced/benchmark-comparison.tsx`

6. Progress Tracking ✅

Location: `lib/video-analysis/comparison/metrics-comparison.ts`

Tracks improvement over time by:
- Fetching user's last 10 video analyses
- Calculating improvement percentage
- Identifying trends (improving/stable/declining)
- Generating personalized insights

Component: `components/video-analysis/enhanced/progress-tracker.tsx`

Displays:
- Score history visualization
- Best/average/latest scores
- Trend indicators
- Progress insights

7. Enhanced Analysis Engine ✅

Location: `lib/video-analysis/enhanced-analysis-engine.ts`

Orchestrates the entire analysis pipeline:

1. **Pose Detection** - Uses TensorFlow.js to detect poses
2. **Technique Analysis** - Extracts pickleball-specific metrics
3. **Visual Overlays** - Generates pose tracking overlays
4. **Drill Recommendations** - Creates personalized training plan
5. **Benchmark Comparison** - Compares to skill level standards
6. **Progress Tracking** - Analyzes historical performance
7. **Coach Commentary** - Generates personalized feedback

8. Database Schema Updates ✅

Updated: `prisma/schema.prisma`

New fields added to `VideoAnalysis` model:

```
visualOverlays         Json?     // Pose tracking overlay data
poseDetectionData      Json?     // Raw pose detection results
drillRecommendations   Json?     // Personalized drill plan
benchmarkComparison    Json?     // Comparison vs benchmarks
progressTracking       Json?     // Historical progress data
totalFramesAnalyzed    Int?      // Number of frames processed
posesDetected          Int?      // Number of poses detected
usedTensorFlow         Boolean   @default(false)
```

9. API Routes ✅

New Route: `app/api/video-analysis/analyze-enhanced/route.ts`

- Accepts video upload
- Performs TensorFlow.js analysis
- Returns comprehensive results
- Saves to database
- Max duration: 300 seconds (5 minutes)

Usage

Backend API Call

```
const response = await fetch('/api/video-analysis/analyze-enhanced', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    videoId: 'video_id_here',
    videoUrl: '/uploads/videos/video.mp4',
    skillLevel: 'INTERMEDIATE',
    useTensorFlow: true
  })
});

const result = await response.json();
```

Frontend Components

```
import {
  PoseOverlayViewer,
  DrillRecommendations,
  BenchmarkComparison,
  ProgressTracker
} from '@/components/video-analysis/enhanced';

// In your component
<PoseOverlayViewer
  videoUrl={analysis.videoUrl}
  overlays={analysis.visualOverlays}
/>

<DrillRecommendations
  recommendations={analysis.drillRecommendations}
/>

<BenchmarkComparison
  comparison={analysis.benchmarkComparison}
  skillLevel="intermediate"
/>

<ProgressTracker
  progress={analysis.progressTracking}
/>
```

Performance Considerations

1. **Frame Sampling**: Analyzes every 5th frame to balance accuracy and speed
2. **Model Selection**: Uses MoveNet Lightning (fastest model) for real-time processing
3. **Resource Cleanup**: Automatically disposes TensorFlow resources after analysis
4. **Async Processing**: Runs analysis in background with progress updates
5. **Database Optimization**: Stores compressed JSON data for overlays

Testing

Manual Testing Checklist

1. ✅ Upload a pickleball video (MP4, MOV, or AVI)
2. ✅ Trigger enhanced analysis via API
3. ✅ Verify pose detection works (check console logs)
4. ✅ Confirm overlays are generated
5. ✅ Validate drill recommendations appear
6. ✅ Check benchmark comparison displays
7. ✅ Ensure progress tracking works with multiple videos
8. ✅ Test visual overlay viewer (play/pause, toggle overlays)

Expected Processing Time

- **2-5 min video**: 1-2 minutes
- **5-10 min video**: 3-5 minutes
- **10-15 min video**: 5-8 minutes

Deployment

Environment Requirements

- Node.js 18+ with TensorFlow.js support
- ffmpeg installed on server (for frame extraction)
- Canvas library (already installed)
- Sufficient RAM for TensorFlow model (512MB+)

Build Steps

```
# 1. Generate Prisma client with new schema
npx prisma generate

# 2. Push schema changes to database (if needed)
npx prisma db push

# 3. Build Next.js app
npm run build

# 4. Start production server
npm run start
```

Verifying Deployment

1. Check TensorFlow.js initialization in logs

2. Test video upload and analysis
3. Verify pose detection data in database
4. Confirm UI components render correctly

Troubleshooting

Issue: TensorFlow.js fails to load

Solution:
- Verify `@tensorflow-models/pose-detection` and `@tensorflow/tfjs-node` are installed
- Check Node.js version (18+)
- Ensure sufficient memory allocated

Issue: Frame extraction fails

Solution:
- Install ffmpeg: `apt-get install ffmpeg`
- Verify video file format is supported
- Check file permissions on temp directory

Issue: Analysis times out

Solution:
- Reduce video length or resolution
- Increase API timeout (currently 300s)
- Use QUICK analysis mode instead of FULL

Issue: Overlays not displaying

Solution:
- Check `visualOverlays` data exists in database
- Verify video player component is rendering
- Check browser console for errors

Future Enhancements

Potential improvements for Phase 2:

1. **Real-time Analysis** - Analyze during video upload
2. **Side-by-Side Comparison** - Compare two videos
3. **Pro Player Templates** - Compare against professional players
4. **3D Visualization** - Show technique in 3D
5. **Mobile Optimization** - Faster processing for mobile uploads
6. **Batch Analysis** - Analyze multiple videos at once
7. **Custom Drills** - Allow coaches to create custom drills
8. **Social Sharing** - Share analysis results with friends

Support

For issues or questions:
- Check console logs for TensorFlow errors
- Verify database schema is up to date

- Review API response for error messages
- Test with smaller video files first

Credits

- **TensorFlow.js** - Pose detection model
- **MoveNet** - Fast and accurate pose estimation
- **Canvas** - Overlay rendering
- **Prisma** - Database ORM
- **Next.js** - Full-stack framework