AI Tool Usage Disclosure

FitCoachAR: Real-Time Adaptive Exercise Coaching

Yangyang Zhang Maximilian Fuchs
Seyedmohamad Mirhoseininejad

CAS 772 – Mobile Data Analytics
December 2025

AI Tool Usage Disclosure

In accordance with the course AI Use Policy, this document discloses all uses of AI tools in the preparation of the project report and code.

1 Writing Assistance

Tool: Google Gemini 2.0 Pro (via Cursor IDE)

Purpose: Grammar correction, clarity improvements, and academic style refinement for the Discussion section.

Prompt Used:

```
Correct the grammatical errors of the following
paragraphs and use advanced grammar:

The FormScript feedback system, while effective
for real-time coaching, have several inherent
limitations. First, the current implementation
reliance on manual-defined thresholds...
[full draft text provided]
```

Outcome: The AI suggested corrections for subject-verb agreement, word choice, and sentence structure. All suggestions were reviewed and selectively incorporated. No substantive content was generated by the AI.

1.1 Post-Processing Results Description

Tool: Google Gemini 3 Pro (thinking)

Purpose: Polish a results paragraph for clarity and academic tone.

Prompt Used:

```
Please help polish the following paragraph for
clarity and academic tone, without changing its
technical meaning:

"Table 3 shows that MoveNet is fast but not stable
across views. After adding our post-processing, the
variance becomes much smaller, especially for elbow
and knee joints."
```

Outcome: The AI refined the paragraph to:

> "Table 3 indicates that while MoveNet achieves low inference latency, its predictions exhibit noticeable instability across different viewpoints. After applying the proposed post-processing pipeline, the variance is substantially reduced, particularly for view-sensitive joints such as the elbows and knees."

This polished version was incorporated into the results section.

2 Code Generation

2.1 RepCount Dataset Parser

Tool: Google Gemini 2.0 Pro

Purpose: Generate a Python class to parse the RepCount (LLSP) dataset CSV annotations for FormCode validation experiments.

Prompt Used:

```
Write a Python class to parse the RepCount (LLSP)
dataset CSV annotations. The solution must use
dataclasses to represent a single repetition
(RepAnnotation with start/end frames and index)
and a complete video's annotations (VideoAnnotation
```

```
including exercise type, video name, total rep count,
and a list of RepAnnotation objects).

The main parser class, RepCountDatasetParser, should:
1. Initialize with the dataset's base path
2. Have a parse_split(split: str) method
3. Implement _extract_reps(row) for frame boundaries
4. Implement _find_video_path(video_name, split)
5. Include filter_by_exercise() with name variations
```

Outcome: The AI generated a complete parser implementation using Python dataclasses. The code was reviewed, tested on the actual dataset, and integrated into `dataset_parser.py`. Minor modifications were made for path handling and error cases.

2.2 TikZ Block Diagram

Tool: Google Gemini 2.0 Pro

Purpose: Generate LaTeX TikZ code for the FormScript pipeline visualization.

Prompt Used:

```
Create a TikZ block diagram with:
- Horizontal Flow: Video Frames -> Pose Estimation
  -> FormCodes -> Super FormCodes
- Vertical Drop: Super FormCodes -> Form Analyzer
- Horizontal Flow: Form Analyzer -> LLM -> Feedback
```

Outcome: The AI generated TikZ code which was adapted and integrated into the methodology section figures. Node styling and positioning were manually adjusted.

2.3 EMA Smoothing Function

Tool: Claude Sonnet 4.5 (Anthropic)

Purpose: Generate a simple exponential moving average function for keypoint smoothing.

Prompt Used:

```
I have 2D keypoints from MoveNet in normalized image
coordinates. I want to apply exponential moving average
(EMA) smoothing to reduce jitter, but I want the
smoothing factor to be adjustable. Please generate a
simple Python function for this.
```

Outcome: The AI generated the following function:

```
def ema_smooth(curr, prev, alpha=0.8):
    """
    Exponential Moving Average smoothing.
    curr: current keypoints (N x 2)
    prev: previous smoothed keypoints (N x 2)
    alpha: smoothing factor
    """
    if prev is None:
        return curr
    return alpha * prev + (1 - alpha) * curr
```

This function was integrated into the post-processing pipeline.

2.4 Form Analyzer Debugging

Tool: Google Gemini 2.0 Pro

Purpose: Debug an error in the `form_analyzer.py` module.

Context: The FormAnalyzer class was implemented, but the `categorize_form_code()` method was returning incorrect categories for edge cases.

Prompt Used:

```
I have the following error in my form_analyzer.py:

TypeError: '<' not supported between instances
of 'NoneType' and 'float'

The error occurs in categorize_form_code() when
checking threshold conditions. Here is the method:
```

```
def categorize_form_code(self, exercise, name, value):
    config = FORM_CODES_CONFIG[exercise]
    for cat in config[name]["categories"]:
        if value < cat.get("v_max"):
            return cat["name"]
    return None
```

```
How can I fix this to handle missing v_max values?
```

Outcome: The AI suggested adding a conditional check for `None` values:

```
v_max = cat.get("v_max")
if v_max is not None and value < v_max:
    return cat["name"]
```

This fix was integrated into the existing code. The core logic and architecture of `form_analyzer.py` (419 lines) was developed independently.

3 Verification Statement

All AI-generated code was:

1. Reviewed line-by-line for correctness
2. Tested on actual project data
3. Modified as necessary for integration
4. Verified to produce expected outputs

The author remains fully responsible for all content, code functionality, and any errors in the report.