MLEng Assignment 5: Model Output Filtering & Highlight Reel

Highlight Reel

1 Objectives

By the end of this assignment, you will be able to:

- Design and implement a custom post-processing filter that smooths frame-level predictions while preserving rapid transitions
- Diagnose failure modes by inspecting specific frames where performance is poor
- Replace the F_1 metric with the more flexible F_β family and select an appropriate β
- Tune both model and filter hyperparameters using F_β as the sole optimisation objective
- Produce an updated "highlight reel" for the test portion of your working video

2 Background

Assignment 4 left you with a solid classifier but still too many spurious frame-level activations. The provided `filter_predictions.py` scaffold already contains a basic majority-vote smoother with hysteresis and a duration filter. Your task is to evolve that scaffolding into a production-ready filtering pipeline.

At the same time, the single special case of F_1 may not reflect business priorities. An F_β score allows you to weight recall higher than precision ($\beta > 1$) or vice versa.

F_β definition

$$F_\beta = (1 + \beta^2)\frac{\text{precision} \times \text{recall}}{\beta^2\text{precision} + \text{recall}}$$

3 Part 1: Implement Your Own Filtering Logic

Open `filter_predictions.py` and replace the current two-stage filter with **your** design. Feel free to:

- Chain multiple filters (e.g., median + hysteresis + duration)
- Add adaptive thresholds (e.g., percentage of positives in a sliding window that decays when inactive)
- Leverage temporal context from neighbouring clips

Document your design choices in inline comments and commit the updated script.

4 Part 2: Investigate Failure Frames

1. Run the unfiltered model on the full dataset and compute per-frame errors.
2. Identify the top n frames (or contiguous segments) with the highest loss contribution.
3. Scrub to those timestamps in your video and record *why* the model struggled (e.g., motion blur, occlusion, ambiguous labelling).
4. Summarise your findings in `failure_analysis.md`-one sentence per clip is enough.

5 Part 3: Switch to F_β and Tune

5.1 Choosing β

Plot F_β for $\beta \in \{0.25, 0.5, 1, 2, 4\}$ on a validation split and pick the value that aligns with project goals (hint: for highlight-reel discovery you may prioritise recall). Explain your choice briefly in your submission.

5.2 Implementation

```python
from sklearn.metrics import precision_score, recall_score

def fbeta_score(y_true, y_pred, beta: float = 1.0):
    p = precision_score(y_true, y_pred)
    r = recall_score(y_true, y_pred)
    if p == r == 0:
        return 0.0
    beta_sq = beta ** 2
    return (1 + beta_sq) * p * r / (beta_sq * p + r)
```

Listing 1: Drop-in replacement for the old metric

Update both the classifier hyperparameter grid search and your new filtering optimiser to target this function.

6 Part 4: Generate the Highlight Reel

Using the best model + filter combination on the held-out `test` partition:

- Locate all segments where the filtered output is positive.
- Concatenate those snippets into `highlight_test.mp4`

7 Submission Requirements

1. **GitHub** (same private repo):
   - Updated `filter_predictions.py`
   - Any auxiliary scripts/notebooks used for metric tuning
   - `failure_analysis.md` (bullet list)
2. **At our next workshop** :
   - The F_β vs. β plot (PNG)
   - A 1–2 paragraph rationale for your chosen β
   - The rendered `highlight_test.mp4` (link or upload)
   - Key metrics on the test split (precision, recall, F_β)

8 Tips for Success

- Begin with simple filters and iterate-overfitting the filter will hide genuine errors.
- Keep the optimisation search space small; two or three hyperparameters per stage is enough.
- When in doubt, visualise: raster plots, confusion timelines, histograms of segment lengths.
- Use vectorised NumPy where possible; filtering should be faster than inference.