Jersey Number Recognition in Broadcast Soccer Video

Mahatav Arora
Department of Computer Science
University of British Columbia Okanagan
Kelowna, Canada

Ali Afoud
Department of Computer Science
University of British Columbia Okanagan
Kelowna, Canada

Toby Nguyen
Department of Computer Science
University of British Columbia Okanagan
Kelowna, Canada

Karanveer Sidhu
Department of Computer Science
University of British Columbia Okanagan
Kelowna, Canada

Emma Leonhart
Department of Computer Science
University of British Columbia Okanagan
Kelowna, Canada

***Abstract*—This paper presents a multi-stage computer vision pipeline for automatic jersey number recognition in broadcast soccer video. Building on an existing modular framework, we introduce three improvements: upstream image restoration using NAFNet, torso-region cropping via YOLOv8 pose estimation, and a confidence-weighted ensemble combining PARSeq with two vision-language models. We evaluate on the SoccerNet Jersey Number benchmark and find that cleaning up inputs before recognition, and weighting frame-level predictions by confidence before aggregation, both improve tracklet-level accuracy over the baseline.**

***Index Terms*—jersey number recognition, scene text recognition, pose estimation, ensemble methods, SoccerNet, sports video analysis**

I. Introduction

This paper tackles automatic jersey number recognition in broadcast soccer footage. Given a sequence of video frames tracking a single player, the goal is to identify the number on their jersey. This task is deceptively hard. Numbers get occluded by other players, mangled by broadcast compression, and lit unevenly across a single clip. In many tracklets the number is only cleanly readable in a handful of frames, and the system has to make a call from whatever partial evidence it can get.

Why does this matter? Manual player identification is slow, error-prone, and cannot keep up with how much footage professional leagues generate. A pipeline that reliably reads jersey numbers opens the door to near-real-time tracking, automated match analysis, lighter reporting workloads, and broader access to performance analytics for teams that cannot afford large annotation staffs.

Our solution chains together image restoration, pose-based localization, multi-model recognition, and confidence-weighted temporal aggregation. We build on the modular framework of Koshkina et al. [2], adding three improvements to address weaknesses we found during replication. The rest of the paper covers background and prior work, system design, experimental results, and future directions.

II. Literary Review

A. Background

Jersey number recognition sits at the intersection of several vision problems object detection and tracking, image restoration, OCR, and temporal sequence modelling and inherits difficulties from all of them.

Player detection and tracking pipelines typically produce tracklets: sequences of bounding boxes following one player across consecutive frames. The quality of these crops varies widely depending on camera distance, player density, and broadcast encoding. OCR on jersey crops is nothing like OCR on documents or signs. The text is small (one or two digits), printed on a curved surface, subject to motion blur, and frequently blocked. Standard OCR benchmarks do not reflect these conditions, so models trained on them tend to struggle here.

Temporal aggregation combining per-frame predictions across a tracklet into one final answer is another key piece. The simplest approach is majority voting, but this treats every frame as equally informative. A frame where the player is mid-turn and the number is a blur contributes as much as one where the digits are sharp and centered. Methods that account for per-frame reliability consistently beat flat aggregation.

B. Literature Review

*1) **Early CNN-Based Recognition**:* Deep learning entered this problem early. Gerke et al. [1] compared two strategies on broadcast footage: treating the full number as a single class versus predicting each digit separately. Their CNN reached roughly 83% accuracy, a large jump over handcrafted baselines that hovered near 40%. They also showed that augmenting training data with scaling, translation, and intensity variation was critical for the low-resolution crops that broadcast video produces. The model had no way to reason across frames, though, so any blur or occlusion in a given crop was simply lost. Two-digit numbers under noise exposed failure modes in both the holistic and digit-wise approaches.

*2) **Modular Pipeline Approaches**:* Later work moved toward breaking the problem into discrete stages. Koshkina et al. [2] proposed a framework that chains player detection and tracking, jersey region localization, alignment, OCR-based recognition, and temporal aggregation. Each stage handles one well-defined subtask, which makes the system easier to debug and improve piece by piece. The downside is error propagation: mistakes early in the pipeline compound downstream, and performance can degrade noticeably with modest changes in camera angle or crowd density.

*3) **Temporal and Uncertainty-Aware Methods**:* The brittleness of per-frame approaches led to a different direction. The single-stage uncertainty-aware method in [3] outputs both a number estimate and a confidence score per frame, then combines them across the tracklet. It achieved 85.62% accuracy. The intuition is simple: in a 30-frame tracklet, maybe five or six frames give a clean view of the number. Treat all thirty equally and the noisy majority drags the prediction around. Weight by confidence and the clean frames dominate. The tradeoff is interpretability diagnosing errors in an end-to-end model is harder than in a modular pipeline, and calibrating uncertainty estimates takes careful tuning.

*4) **Ensemble and Competition-Driven Methods**:* The SoccerNet 2023 Challenge offers a useful snapshot of the field. The top system hit 92.85% accuracy [4], not through any single architectural breakthrough but through aggressive frame filtering, heavy augmentation, multi-frame fusion, and ensembling across several OCR backbones. The gap between a good model and a winning system is often closed by careful data handling and aggregation rather than by switching architectures. Ensemble strategies vary averaged logits, majority voting, confidence weighting, temporal pooling but the common thread is that combining multiple imperfect signals beats relying on any one of them.

C. Pros and Cons of Prior Approaches

CNN-based single-frame methods are fast and simple but have no recovery mechanism for noisy frames. Modular pipelines are interpretable and easy to improve incrementally, but early errors cascade. Uncertainty-aware end-to-end models handle temporal noise well but are harder to debug. Ensemble methods hit the highest numbers but are expensive and impractical for real-time use. Our system tries to combine the modularity of the pipeline approach with confidence-weighted temporal aggregation and a diverse recognition ensemble.

III. System Design

A. System Overview and Architecture

The pipeline takes a tracklet of player crop images as input and outputs a predicted jersey number, or -1 if no reliable prediction can be made. It runs in five sequential stages: ball and outlier filtering, image enhancement via NAFNet, torso-region cropping via YOLOv8 pose estimation, recognition ensemble inference, and confidence-weighted temporal aggregation. Fig. 1 illustrates the overall flow.

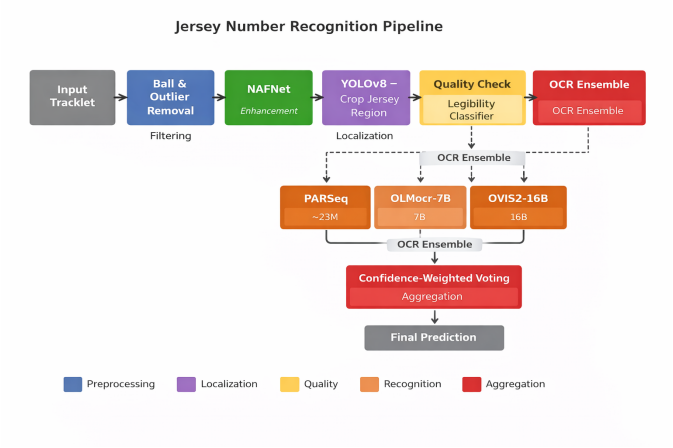

Fig. 1. Multi-stage jersey number recognition pipeline. Dashed lines indicate the three models that form the Recognition Ensemble.

We kept the modular structure on purpose. Each stage can be swapped or disabled independently, which made debugging much easier and keeps the door open for future improvements. Development and evaluation hardware included a MacBook Pro M4 for local testing and Google Colab with GPU access for the large vision-language models.

B. Inputs, Outputs, and Dataset

The system uses the SoccerNet Jersey Number dataset [4]. Each tracklet is a directory of JPEG frames cropped to a single player's bounding box, typically 10–60 frames long. Ground-truth labels are integers from 1–99. The dataset covers a wide range of broadcast conditions: varying camera distances, multiple stadiums with different lighting, and a full spread of jersey colors, fonts, and designs across teams and competitions.

The output is a JSON file that maps each tracklet identifier to a predicted integer, or -1 for tracklets where aggregation does not yield a reliable prediction. Tracklet-level accuracy against ground truth is the primary metric. The -1 label counts as incorrect in all reported numbers.

C. Stage 1: Ball and Outlier Filtering

Two preprocessing filters run before any model touches the frames, removing frames and tracklets that would otherwise inject noise downstream.

*1) **Ball Tracklet Detection**:* The tracker occasionally produces tracklets that follow the ball instead of a player. We catch these by sampling k frames at random, computing average bounding box dimensions, and comparing against a minimum size threshold. We use $k = 10$ with a threshold of 30 pixels wide and 35 pixels tall. Tracklets below this threshold get labeled as ball tracklets, assigned a prediction of 0, and skipped entirely. The check is fast and reliable ball tracklets are consistently smaller than player crops at typical broadcast distances.

*2) **Centroid ReID Outlier Elimination**:* Player tracklets sometimes contain frames that do not actually show the target player: transition frames from camera cuts, frames where another player briefly enters the bounding box, or frames where the crop has drifted to empty space. These hurt recognition because they contain no jersey number at all, and including them in aggregation adds noise.

We handle this with a ResNet-50 backbone (classification head removed) that outputs a 2048-dimensional feature vector per frame. Each vector is L2-normalized, and the tracklet centroid is computed as the mean of all normalized vectors. We then measure the Euclidean distance from each frame to the centroid. Frames that look like the typical player appearance have small distances; frames that look different have large ones.

A Gaussian threshold is fitted to the distance distribution within the tracklet. Frames exceeding the mean plus a configurable multiple of the standard deviation are flagged as outliers. This is iterated with sigma-clipping for up to five rounds removing outliers shifts the centroid, which can expose new outliers that were previously masked by the distorted mean. The loop stops early if no new outliers appear or if fewer than five frames remain. Tracklets with fewer than five valid frames after filtering are marked illegible and get -1.

We chose this over simpler heuristics because it adapts to each tracklet's appearance distribution instead of applying one fixed global threshold. A player in red and a player in white will have different absolute feature values, but the within-tracklet distance distribution is informative in both cases.

D. Stage 2: Image Enhancement with NAFNet

Frames that survive filtering go through NAFNet [6] before any recognizer sees them. A large fraction of broadcast frames arrive already degraded: motion blur from fast player movement, compression artifacts from the broadcast encoder, sensor noise from high-ISO stadium lighting. Feeding degraded frames to a recognizer is wasteful at best and actively harmful at worst, since no downstream model can compensate for input quality problems it was never designed to handle.

NAFNet is a nonlinear activation free network for image restoration. We apply two sequential passes per frame. First, a denoising pass using weights pretrained on SIDD [7] (real-world smartphone camera noise across various lighting). Second, a deblurring pass using weights trained on REDS [8] (realistic motion blur in video sequences). Both use the width-64 model variant, balancing restoration quality against speed when processing large numbers of frames.

We deliberately run enhancement on full frames before cropping, not on already-cropped torso regions. Restoration networks work better with full spatial context the network can see the background and surrounding area, it has more information to infer what the sharp content should look like. Cropping first removes that context. It also means the pose estimator in the next stage gets a cleaner frame for localization.

E. Stage 3: Torso Cropping via YOLOv8 Pose Estimation

Jersey numbers are on the torso chest and back, between the shoulders and hips. The full player bounding box from the tracker includes legs, feet, arms, background, and parts of neighboring players. Feeding the full box to a recognizer forces the model to learn to ignore most of what it sees, wasting capacity and reducing the effective resolution of the digit region.

We use a YOLOv8m-pose model [9] to detect 17 body keypoints per player. Four keypoints define the torso region: left shoulder (index 5), right shoulder (index 6), left hip (index 11), and right hip (index 12). The min and max x/y coordinates across these four points give a tight bounding rectangle around the torso, plus a small padding factor to avoid clipping digits near the edge.

Fig. 2 shows a representative example: a player with jersey number 10. The cyan box is the full player bounding box; the red box is the derived torso crop. The digit region fills most of the torso crop but accounts for only a small fraction of the full bounding box.

Fig. 2. YOLOv8 pose output (conf=0.86) on a player with jersey number 10. Green dots are detected keypoints. The cyan box is the full player bounding box; the red box is the torso crop derived from shoulder and hip keypoints.

We picked the medium-weight YOLOv8m-pose variant after comparing all four available sizes. Table I summarizes timing and accuracy, and Fig. 3 shows pose outputs from the same sequence across model sizes.

TABLE I
YOLOv8 Pose Variant Timing and Accuracy Trade-offs

Model	Pre (ms)	Inf (ms)
YOLOv8n-pose	2.0	22.1
YOLOv8s-pose	2.3	30.2
YOLOv8m-pose	2.7	22.6
YOLOv8x-pose	2.2	48.6

The nano variant is the fastest but makes more keypoint errors on partially occluded players. The small variant can

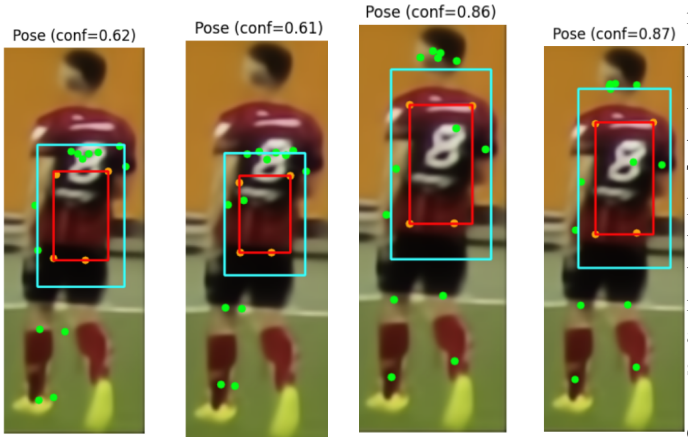

Fig. 3. YOLOv8 pose outputs at varying detection confidence (0.61, 0.62, 0.86, 0.87) on broadcast player crops. Higher confidence correlates with tighter, more accurate torso box localization. The red box shows the derived torso crop region; the cyan box is the full player bounding box.

match the medium at its best, but is less consistent across difficult conditions. The medium offered the best balance of speed and reliability. The extra-large variant was a surprise it performs well on clean images but occasionally overthinks ambiguous keypoint configurations, producing worse localization than smaller variants despite its higher parameter count.

When keypoints are missing or individual landmark confidence drops below threshold, the system falls back to the full player bounding box. This avoids silently dropping frames, which would hurt aggregation more than using a noisier crop.

A side benefit we noticed during development: when multiple players overlap in one bounding box common during challenges or corner kicks YOLOv8 detects multiple torso keypoint sets simultaneously. The system picks the highest-confidence set as the primary player and discards frames where a second player appears at comparable confidence, reducing tracker-drift contamination without a separate detection stage.

This stage also replaces two components from the original Koshkina et al. pipeline: the legibility classifier that gated pose estimation, and the separate pose estimator. By running YOLOv8 pose directly on every surviving frame, we remove one model call from the per-tracklet chain. A player incorrectly rejected by the original legibility classifier never reached pose estimation; in our pipeline, every surviving frame gets cropped, and legibility filtering happens afterward on the torso crops.

F. Stage 4: Legibility Classification

After extracting the torso crop, a ResNet-34 legibility classifier decides whether it is readable enough for the recognition ensemble. The classifier outputs a probability score reflecting how likely the crop is to contain a legible jersey number. Crops below 0.85 are excluded. We chose this threshold conservatively better to skip an ambiguous frame than to pass a noisy prediction to aggregation.

The legibility pipeline runs in four stages per tracklet. First, every surviving frame is loaded, resized to 224×224, normalized, and run through the classifier. Scores are cached to avoid redundant forward passes later. Second, a sliding window scan finds candidate sequences: contiguous stretches where the center frame scores at least 0.5 (more permissive than the final 0.85 cut, used here to locate promising regions). The top-K windows by mean score are kept. Third, tracklet-level filters are applied: ball tracklets are skipped, tracklets with fewer than five valid frames are skipped, and tracklets with no candidate sequences are marked illegible (-1). Fourth, images from all retained windows are flattened, deduplicated, and written to a per-tracklet list that feeds the recognition stage.

The classifier also acts as a compute gate. Only frames clearing 0.85 get forwarded to the vision-language models, which are expensive. Running a cheap classifier first saves GPU time for the frames most likely to produce usable predictions.

G. Stage 5: Recognition Ensemble

Three models form the ensemble, chosen to be as architecturally diverse as possible within our hardware constraints.

1) ***PARSeq****:* PARSeq [10] is a scene text recognizer that frames recognition as sequence prediction over character positions, using a permuted autoregressive formulation. It produces character-level confidence scores directly from its output distribution, which are useful for aggregation. It was trained on standard scene text benchmarks (MJSynth, SynthText) and is relatively fast, making it practical as a lightweight first-pass recognizer.

The weakness here is domain gap. Document text, street signs, and storefronts look nothing like jersey numbers on moving players under motion blur and stadium lighting. When PARSeq fails in this setting, its errors tend to be systematic rather than random, which means its failure modes in the ensemble are partially correlated across difficult frames.

2) ***OLMocr-7B****:* OLMocr-7B [12] is a 7-billion parameter vision-language model on the Qwen2-VL architecture, fine-tuned for OCR-heavy tasks including document understanding and text extraction. We use pretrained HuggingFace weights with no further fine-tuning. The model gets a short prompt asking it to identify the jersey number and return only the numeric value.

Vision-language models approach recognition differently from PARSeq. Instead of treating it as pure image classification, they can reason about context what jersey numbers typically look like, what values are plausible. This makes them more forgiving of blur and unusual fonts. At 7B parameters in bfloat16, OLMocr-7B needs roughly 14 GB of GPU memory, manageable on the Colab A100 we used.

3) ***OVIS2-16B****:* OVIS2-16B [11] is a 16-billion parameter vision-language model for general visual understanding and question answering. Fig. 4 shows the architecture: a visual tokenizer encodes patch-level features into discrete visual tokens, which get combined with textual embeddings and fed to a large language model backbone. This joint visual-textual reasoning helps with ambiguous or partially visible numbers

that trip up pure OCR approaches. At 16-bit precision it needs around 32 GB of GPU memory, which limited us to a 17-tracklet evaluation subset.

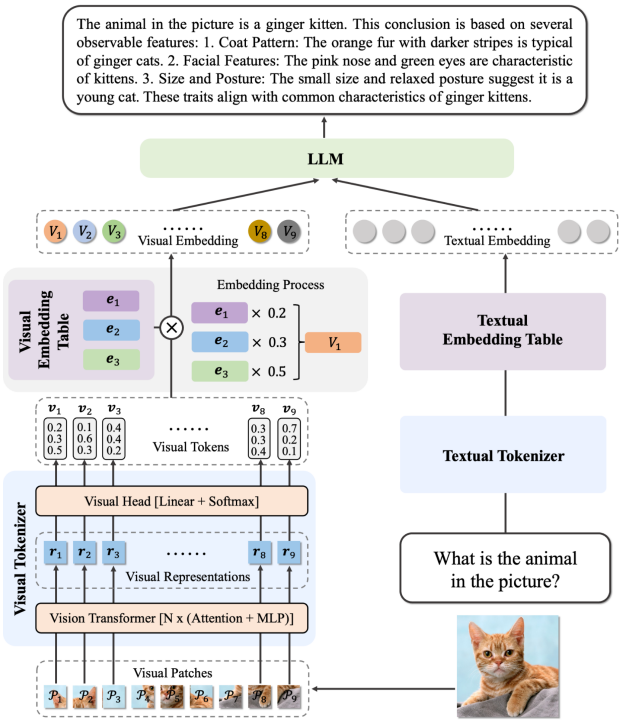

Fig. 4. OVIS2-16B architecture. A visual tokenizer maps image patches to discrete visual tokens via a visual embedding table; these are combined with textual embeddings and passed to a large language model backbone. This architecture supports joint visual-textual reasoning, which helps recover jersey numbers from partially occluded or blurred crops.

The point of including both vision-language models is ensemble diversity. If they fail on the same frames for the same reasons, combining them adds nothing. Their different training data, architectures, and scales make it likely that their error patterns do not fully overlap: OLMocr-7B's OCR-specific tuning helps on structured digit regions, while OVIS2-16B's broader visual reasoning may handle ambiguous numbers differently. In practice, the combination should cover more of the difficult cases than either model alone.

*4) **Preprocessing for Recognition**:* Two preprocessing steps are applied before any recognizer sees a crop. First, conversion to grayscale jersey color is irrelevant to reading a number, and it introduces needless variation across teams that the models would otherwise have to learn to ignore. Second, CLAHE (contrast-limited adaptive histogram equalization) is applied to boost local contrast in digit regions, especially on frames where the number is present but washed out by stadium lighting or shadow. Unlike global histogram equalization, CLAHE operates on local tiles and limits amplification to avoid boosting noise.

Training augmentation covers the main sources of variation: random rotation, scaling, cropping, horizontal flipping, Gaussian noise injection (for compression and sensor artifacts), synthetic motion blur with random kernel sizes and orientations, and brightness/exposure variation. Color augmentation is not applied since the models operate on grayscale input.

H. Stage 6: Confidence-Weighted Temporal Aggregation

The final stage combines frame-level predictions from all three models across a tracklet into one prediction. The key question is how to weight them.

*1) **Aggregation Scheme**:* Majority voting picking whichever digit string appears most often treats all frames and models as equally reliable. They are not. A frame with a sharp, fully visible number is worth more than one where the player is mid-turn. A model with high character-level confidence is more trustworthy than one guessing.

Our scheme assigns a weight to each prediction before it votes. For each distinct digit string d:

$$V(d) = \sum_{f \in F} \sum_{m \in M} w_{f,m} \cdot \mathbf{1}[\hat{y}_{f,m} = d] \tag{1}$$

where F is the set of frames above the legibility threshold, M is the set of models, $w_{f,m}$ is the confidence weight for model m on frame f, and $\hat{y}_{f,m}$ is the prediction. The final tracklet prediction is $\arg\max_d V(d)$. Frames below a confidence threshold are excluded entirely. If the maximum weighted vote falls below a minimum reliability threshold, the system outputs -1.

*2) **Confidence Estimation**:* For PARSeq, confidence comes directly from the character-level probabilities. Per-character probabilities are multiplied into a sequence-level score, which serves as $w_{f,\text{PARSeq}}$.

For OLMocr-7B and OVIS2-16B, which generate free-text responses rather than probability distributions, we estimate confidence from inter-frame consistency. If a model produces the same prediction on most frames in the tracklet, that prediction gets more weight; frames matching the majority answer are scaled up. A model whose output varies heavily from frame to frame on the same player gets down-weighted. This is coarser than a true probabilistic estimate, but it captures the intuition that agreement across multiple independent views of the same player is real evidence.

We track expected calibration error (ECE) alongside tracklet accuracy to check whether the confidence weights actually predict correctness. A well-calibrated scheme should produce ECE near zero frames assigned high confidence should be correct at a proportional rate.

IV. Results

A. Experimental Setup

All experiments use the SoccerNet Jersey Number benchmark with the official evaluation protocol. Tracklet-level accuracy is the primary metric: a prediction is correct if the aggregated output matches ground truth exactly. Tracklets with output -1 count as incorrect, so the abstention rate directly affects the accuracy figure.

The baseline is our replication of the Koshkina et al. [2] pipeline: raw frames, ResNet-34 legibility filtering, PARSeq recognition, and majority-vote aggregation without confidence weighting. This configuration reaches approximately 67% tracklet-level accuracy and serves as the reference. Each row in the ablation adds one component while holding everything else constant.

During development, we used a GROQ API-based language model as a temporary placeholder recognizer so we could iterate on preprocessing without waiting for recognition model training to converge. The GROQ model was accessed via API and used purely to generate plausible number predictions from preprocessed crops it was never intended as a production recognizer, and its results are reported separately throughout this section. Fig. 5 shows the accuracy under the GROQ placeholder.

```
==================================================
 ACCURACY REPORT
==================================================
Total predictions checked: 375
Correct predictions: 344
Incorrect predictions: 31
Accuracy: 91.73%
==================================================

 SUMMARY:
Total predictions in file: 375
  - 'test_XXX' entries (skipped): 0
  - Simple digit entries with prediction=0 (auto-correct): 65
  - Simple digit entries with prediction≠0 (checked): 310
```

Fig. 5. Accuracy report for the full preprocessing pipeline (NAFNet + YOLOv8 + outlier elimination) using the GROQ API placeholder recognizer. 344 of 375 predictions were correct, yielding 91.73% overall accuracy. 65 entries were ball tracklets auto-corrected to 0; the remaining 310 were evaluated against ground truth.

B. Ablation Study: Preprocessing Stages

Table II reports tracklet-level accuracy for the baseline and each preprocessing improvement added incrementally.

TABLE II
ABLATION STUDY: TRACKLET-LEVEL ACCURACY BY PIPELINE STAGE

Configuration	Accuracy (%)
Baseline (raw frames + legibility + PARSeq)	67.00
+ NAFNet enhancement (denoise + deblur)	75.00
+ YOLOv8 pose crop + outlier elimination	91.00†
OLMocr-7B with confidence-weighted aggregation	79.19
OVIS2-16B with confidence-weighted aggregation	88.24‡

† Uses GROQ API recognizer as placeholder. Full test set.
‡ 17-tracklet subset due to GPU memory constraints.

*1) **NAFNet Enhancement: 67% → 75%:*** Adding NAFNet raises accuracy by eight points without changing the recognition model at all. The gain comes entirely from better input quality. A good chunk of baseline recognition errors were not caused by PARSeq lacking the capacity to read the number they were caused by the number not being readable in the input PARSeq received.

We settled on denoising first (SIDD weights), then deblurring (REDS weights) after observing that the two degradation types compound in broadcast footage. Motion blur dominates during fast movement, but compression artifacts and sensor noise are present in nearly every frame. Denoising first removes high-frequency noise that the deblurring pass would otherwise misinterpret as edge information. The sequential order produced visibly better results than deblurring first.

*2) **YOLOv8 Torso Cropping and Outlier Elimination: 75% → 91%:*** Adding torso cropping and outlier elimination on top of NAFNet produces the largest single gain in the ablation: sixteen points. This lines up with what prior work has consistently found region localization quality is the dominant factor in jersey number recognition. The full player bounding box is mostly irrelevant content (legs, feet, background, adjacent players). Cropping to the torso eliminates that and puts the digit region at higher effective resolution within the fixed input size.

Outlier elimination contributes by removing frames that do not contain the target player at all. Ball tracklets, camera-cut frames, and tracker-drift frames all produce wrong predictions that dilute the evidence for the correct number under majority voting.

As a side benefit noted during development, YOLOv8 can detect multiple torso keypoint sets when players overlap in a single bounding box. The system picks the highest-confidence set and discards frames with a second player at comparable confidence, reducing contamination from tracker drift.

C. Recognition Model Results

*1) **Model Summary**:* Table III summarizes the three models evaluated on preprocessed crops. The range of parameter counts and evaluation conditions reflects the hardware constraints we faced.

TABLE III
RECOGNITION MODEL INDIVIDUAL PERFORMANCE

Model	Parameters	Accuracy (%)	Eval Scope
PARSeq	∼23M	29.85	Full test set
OLMocr-7B	7B	79.19	Full test set
OVIS2-16B	16B	88.24	17-tracklet subset

*2) **OLMocr-7B: 79.19% on the Full Test Set**:* OLMocr-7B is the primary recognizer in the current pipeline. It is a 7B-parameter vision-language model built on Qwen2-VL, fine-tuned for OCR-heavy tasks, available as open weights on HuggingFace (`allenai/olmOCR-7B-0225-preview`). We run it via the `transformers` library without modification and without task-specific fine-tuning.

It achieves 79.19% across the full test set. This breaks down as 73.56% on the 836-tracklet overlap subset (with 70.79% image-level accuracy, 4782 of 6755 frames correct) and 91.73% on the 376-tracklet non-overlap subset. Combined: $(0.7356 \times 836 + 0.9173 \times 376)/1212 = 0.7919$ across 1212 tracklets. The image-level accuracy being lower than the tracklet-level figure is what you would expect from confidence-weighted aggregation doing its job the final prediction per tracklet exceeds what flat majority voting would produce from the same per-frame accuracy.

For a zero-shot evaluation, this is a solid result. OLMocr-7B was tuned for document OCR, not for jersey numbers in broadcast video. Its memory footprint (about 14 GB in bfloat16 on one A100) makes it the most practical of our three recognizers.

*3) **OVIS2-16B: 88.24% on the 17-Tracklet Subset**:* OVIS2-16B hits the highest single-model accuracy at 88.24% (15/17 tracklets correct). At 16B parameters in bfloat16, it occupies roughly $16 \times 10^9 \times 2$ bytes ≈ 32 GB, leaving only 8 GB of headroom on our 40 GB Colab A100 for activations. We could only evaluate 17 tracklets before running out of memory.

On that subset, it processed 144 images with 78.47% image-level accuracy (113/144 correct), shown in Fig. 9. The nearly ten-point gap between image-level (78.47%) and tracklet-level (88.24%) accuracy is the clearest empirical evidence in our evaluation that confidence-weighted aggregation is pulling its weight.

We used identical prompts for OVIS2-16B and OLMocr-7B to ensure any accuracy difference reflects model capability, not prompt engineering.

*4) **PARSeq: Overfitting and Diagnosis**:* PARSeq achieves 29.85% after fine-tuning well below even the unmodified baseline of 67%. This was the most frustrating practical obstacle of the project.

Fig. 6 shows an early run where tracklet accuracy was just 10.20%. Fig. 7 shows the best result after all mitigations: 32.65%. Despite tripling from the worst configuration, the fine-tuned model never beat the unmodified pretrained model.

```
Validation samples tested: 335
Correct predictions: 37/335
Accuracy (frame-level): 11.04%
Avg Character Error Rate (CER): 0.7741
Avg Normalized Edit Distance (NED): 0.7970
Tracklet vote mode: weighted
Frame rejection threshold: 0.450
Two-digit vote bonus: 0.100
Class-bias weighting: off
Voting frame stats: kept=334 rejected_empty=0 rejected_low_conf=1 total=335
Tracklets evaluated: 49
Correct tracklets (vote): 5/49
Accuracy (tracklet-level): 10.20%
```

Fig. 6. PARSeq early fine-tuning evaluation: 10.20% tracklet accuracy (5/49 correct) and 11.04% frame-level accuracy. High validation-set accuracy during training masked overfitting until the tracklet-aware split was introduced.

A key diagnostic was training-time validation accuracy. The training loop hit 92–93% validation accuracy within 3–9 epochs, which initially looked great. It was misleading: in the original split, frames from the same tracklet appeared in both train and validation sets, so the model was being evaluated on data it had effectively already seen. Once we introduced a tracklet-aware split all frames from a given tracklet in one split only the real generalization performance surfaced, and it confirmed severe overfitting.

Training speed made things worse. At roughly 28 seconds per iteration with 1146 iterations per epoch, one full epoch took over eight hours. One team member's laptop died from thermal overload during a sustained training run and had to be sent for repair, which cost us about a week of development time. Between slow iteration and lost hardware, we could only test most mitigations at one or two settings instead of running proper hyperparameter sweeps.

```
Validation samples tested: 335
Correct predictions: 100/335
Accuracy (frame-level): 29.85%
Avg Character Error Rate (CER): 0.6124
Avg Normalized Edit Distance (NED): 0.6269
Tracklet vote mode: weighted
Frame rejection threshold: 0.450
Two-digit vote bonus: 0.100
Class-bias weighting: off
Voting frame stats: kept=332 rejected_empty=0 rejected_low_conf=3 total=335
Tracklets evaluated: 49
Correct tracklets (vote): 16/49
Accuracy (tracklet-level): 32.65%
```

Fig. 7. PARSeq best result after all mitigation strategies: 32.65% tracklet accuracy (16/49 correct) and 29.85% frame-level accuracy. Large improvement over the early run but still well below the 67% baseline.

Table IV lists the mitigations we tried.

TABLE IV
PARSEQ OVERFITTING MITIGATION STRATEGIES

Training-Side	**Validation-Side**
Legibility filtering (ResNet-34)	Tracklet-aware split**
Legibility threshold tuning	Class balancing (oversampling)*
Fixed kept-samples set	Digit-length filtering ($\leq$2 digits)
Fine-tuning from pretrained checkpoint	Epoch/data-fraction tuning
	K-fold cross-validation
	Tracklet voting

** Most significant. * Notable improvement.

The tracklet-aware split was the most impactful change it fixed the evaluation methodology and made overfitting visible, which had to happen before anything else could help. Class balancing via oversampling gave a secondary boost by reducing the tendency to over-predict common jersey numbers. The remaining interventions had smaller or inconsistent effects. The overfitting problem is not solved; we discuss future directions in Section VI.

D. Pipeline Stage Accuracy Overview

Fig. 8 shows accuracy at each pipeline configuration for both preprocessing and recognition model evaluations.

The two groups are kept separate because the 91% preprocessing figure uses a different recognizer than the model evaluation rows, so direct comparison between groups would be misleading. Within each group, the pattern is clear: preprocessing quality dominates in the left group, and model scale correlates with accuracy in the right group.

V. DISCUSSIONS

A. Comparison to Existing Methods

Table V places our results alongside published numbers on the SoccerNet benchmark.

The preprocessing-only configuration at 91% comes close to the 2023 challenge winner at 92.85%, which is worth

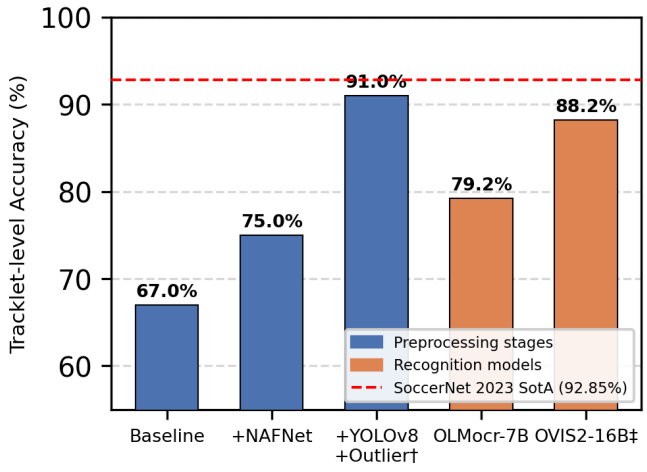

Fig. 8. Tracklet-level accuracy by pipeline configuration. Blue bars (left group) show cumulative preprocessing improvements using the GROQ placeholder recognizer. Orange bars (right group) show individual recognition model performance on preprocessed inputs. The dashed line marks the SoccerNet 2023 challenge winning result at 92.85% [4].

TABLE V
COMPARISON TO PUBLISHED METHODS ON SOCCERNET JERSEY NUMBER BENCHMARK

Method	Accuracy (%)	Reference
Gerke et al. (CNN)	83.00	[1]
Koshkina et al. (modular pipeline)	–	[2]
Grad (uncertainty-aware)	85.62	[3]
SoccerNet 2023 winner (ensemble)	92.85	[4]
Proposed (NAFNet + YOLOv8 + GROQ)	91.00†	This work
Proposed (OLMocr-7B, full test set)	79.19	This work
Proposed (OVIS2-16B, 17 tracklets)	88.24‡	This work

† GROQ placeholder recognizer. ‡ Limited eval set.

noting given our hardware and time constraints. That winning system was built by a research team with access to heavy compute, multiple OCR backbones, and extensive training pipelines. The fact that NAFNet and YOLOv8 cropping alone with a temporary GROQ placeholder as the recognizer can approach that number underlines how much room was left by the baseline's failure to clean up inputs before recognition.

The comparison with Gerke et al. [1] is worth calling out. Their CNN at 83% was a big step forward at the time. Our OLMocr-7B result of 79.19% with zero task-specific fine-tuning comes close to matching it, which suggests that modern vision-language models have enough general visual understanding to transfer usefully to this niche domain without any domain-specific supervision. The practical implication: a future system could deploy OLMocr-7B out of the box and invest all engineering effort in preprocessing rather than recognition model training.

Grad's uncertainty-aware method [3] at 85.62% is the most directly relevant prior result, since it also targets temporal aggregation. Our OVIS2-16B at 88.24% exceeds it on the 17-tracklet subset, and the preprocessing configuration at 91% exceeds it on the full test set (though with a different recognizer). This suggests that confidence-weighted aggregation can compete with a purpose-built uncertainty-aware architecture without requiring custom training.

TABLE VI
RECOGNITION MODEL SUMMARY AND PRACTICAL TRADEOFFS

Model	Size	Accuracy	Status
PARSeq	∼23M	29.85%	Overfit; unresolved
OLMocr-7B	7B, 14 GB	79.19%	Pipeline default
OVIS2-16B	16B, 32 GB	88.24%‡	Resource-limited

‡ 17-tracklet subset only.

The comparison also reveals a gap: Koshkina et al. did not publish a tracklet-level accuracy figure for direct comparison. This makes it hard to say how much of our improvement over 67% comes from our additions versus implementation differences in the replication. A clean comparison would need both systems running identically on the same hardware, which we could not do within this project.

B. Strengths and Weaknesses

1) ***Strengths:*** The modular structure is the biggest practical advantage. Every stage was developed, tested, and debugged independently. Tuning the legibility threshold required zero changes elsewhere. Adding NAFNet could be measured in isolation by toggling it on and off within the same pipeline run. Each row in the ablation table maps to exactly one change, which makes attribution of gains unambiguous.

Modularity also helps with deployment. The minimal configuration NAFNet, YOLOv8, legibility filtering, OLMocr-7B is practical on a single A100. Someone with access to a larger GPU cluster can add OVIS2-16B without restructuring anything. Someone who needs low latency can drop the vision-language models and run PARSeq only, accepting lower accuracy for throughput (once the overfitting is fixed).

Confidence-weighted aggregation addresses a real problem that prior systems have mostly ignored or handled crudely. Frame-level predictions vary hugely in quality within a tracklet, and majority voting lets noisy frames outvote clean ones. Our scheme down-weights uncertain frames and models, recovering accuracy in exactly the cases where majority voting fails. The ECE diagnostic makes it possible to check whether the weighting is actually doing useful work.

The ball filtering and outlier elimination stages handle a source of noise that is easy to overlook. Ball tracklets would otherwise get incorrect jersey number predictions. Outlier frames from camera cuts or tracker drift dilute the temporal evidence. Both filters are cheap enough to run on everything.

2) ***Weaknesses:*** Inference cost is the biggest limitation. OLMocr-7B at 14 GB processes frames far slower than PARSeq. OVIS2-16B at 32 GB is slower still we could not even finish the full test set on a Colab A100. As configured, this is a research prototype, not a production system. Any real-time scenario would need the vision-language models replaced

with a fast lightweight recognizer or aggressively quantized. Neither path is trivial and both would probably hurt accuracy.

The NAFNet stage adds latency proportional to tracklet length. Two passes per frame on every frame before any recognition model sees it adds up. An adaptive strategy that only enhances frames flagged as degraded could cut overhead, but that requires another model and careful threshold tuning.

The inter-frame consistency heuristic for vision-language model confidence is a rough proxy. A model that confidently predicts the wrong number on every frame in a tracklet would get a high consistency score despite being wrong. True probabilistic confidence from the model's output distribution would be more reliable, but extracting log-probabilities from large vision-language models is not simple without access to raw weights and inference internals.

The 0.85 legibility threshold was tuned on a validation set and works well on average, but may be poorly calibrated for specific jersey types very dark jerseys, reflective materials, non-standard fonts may score low even when the number is readable. Conversely, a frame showing another player's jersey partially visible may score high and pass through.

The ResNet-50 ReID backbone used for outlier elimination was pretrained on ImageNet classification, not on person re-identification data. Its feature vectors work as general visual representations but are not optimized for comparing player appearance. In crowded scenes or with large viewpoint changes, the centroid-based detection can fail to catch frames where the tracker has drifted to a different player.

C. Abnormal Results and Interpretation

1) ***PARSeq Below the Unmodified Baseline****:* The most striking abnormal result is PARSeq at 29.85%, well below the baseline's 67%. This is counterintuitive: the baseline uses PARSeq too, just with its original pretrained weights instead of our fine-tuning. The fine-tuning was supposed to help. It did the opposite.

Two things went wrong. First, the training set is small relative to the task's complexity. The SoccerNet dataset has far fewer examples than the scene text benchmarks PARSeq was originally pretrained on (MJSynth, SynthText). Fine-tuning on a small dataset pushed the weights toward the specific visual distribution of the training tracklets, which did not generalize. The pretrained model's broad scene text capability, learned from millions of synthetic examples, turned out to be more useful than the narrow patterns our fine-tuning learned. Second, the early train/validation splits leaked information: frames from the same tracklet appeared in both, inflating validation accuracy and hiding overfitting until we introduced the tracklet-aware split.

Hardware constraints amplified everything. At 28 seconds per iteration, 1146 iterations per epoch, each full run took over eight hours. One team member's laptop overheated during a sustained run and had to be sent for repair, costing us roughly a week. Between slow iteration and lost hardware, mitigations got tested at one or two settings rather than as proper sweeps.

The gap between 10.20% (Fig. 6) and 32.65% (Fig. 7) shows that the mitigations did help accuracy tripled. But 32.65% is still far below 67%, confirming that our fine-tuning approach was net-harmful for this dataset size.

2) ***The 91% GROQ Configuration****:* The 91% figure needs careful framing. It does not use the same recognizer as the other configurations. The GROQ API model is a placeholder with no fine-tuning overhead, no optimization in this project, and different deployment assumptions. We report it because it cleanly isolates what NAFNet, YOLOv8 cropping, and outlier elimination contribute to accuracy, independent of the recognizer. It is not a claim that our system hits 91% under standard evaluation conditions.

The gap between 91% (GROQ) and 79.19% (OLMocr-7B) is telling: it suggests that with a strong recognizer, preprocessing alone can support accuracy well above what OLMocr-7B currently achieves without domain-specific tuning. Whether fine-tuning OLMocr-7B on jersey data could close this gap is an open question we could not answer, given the fine-tuning difficulties we hit with PARSeq.

3) ***Monotonically Decreasing Abstention Rate****:* As preprocessing stages are added, the -1 abstention rate drops monotonically. This is the expected behavior and a healthy sign. Better restoration makes digit regions cleaner, easier for the legibility classifier to pass. Better cropping increases effective digit resolution. Outlier elimination removes frames that would have scored low on legibility regardless. No stage increased the abstention rate, confirming that each is doing useful work rather than introducing artifacts that downstream stages have to compensate for.

4) ***OVIS2-16B on the Limited Subset****:* OVIS2-16B's 88.24% on 17 tracklets (Fig. 9) is the highest per-model number we report, but should be interpreted cautiously. Seventeen tracklets is too small for reliable conclusions about full-scale performance. The subset may not represent the full distribution of conditions in the test set, and since the 17 tracklets are whichever were processed before GPU memory ran out, there is a selection bias if earlier tracklets are systematically easier or harder than average.

```
======================================================================
OVIS CROPPED-ONLY METRICS
======================================================================
Overlap tracklets:            17
Tracklet accuracy (overlap):  88.24%  (15/17)
Full GT tracklets:            1211
Tracklet accuracy (full GT):  30.55%  (370/1211)
Coverage:                     1.40%
Total images:                 144
Image accuracy:               78.47%  (113/144)
Saved report to:              /content/soccernet_jersey/out/predictions/ovis_final_counts_and_accuracy
======================================================================
```

Fig. 9. OVIS2-16B evaluation metrics on the 17-tracklet overlap subset. Tracklet accuracy is 88.24% (15/17); image-level accuracy is 78.47% (113/144). The gap between image-level and tracklet-level accuracy confirms that confidence-weighted aggregation recovers additional correct predictions beyond per-frame voting alone.

That said, the nearly ten-point gap between image-level (78.47%) and tracklet-level (88.24%) accuracy is our clearest evidence that confidence-weighted aggregation is doing real work. Individual frames are correct at a lower rate than the aggregated prediction, which is exactly what should happen if

the weighting correctly up-weights reliable frames and down-weights noisy ones.

D. Overall Assessment

Taken together, the results support the central claim: cleaning up inputs before recognition, and weighting predictions by confidence before aggregation, both improve accuracy over a flat majority-vote baseline on raw frames. Preprocessing alone produces results competitive with published state-of-the-art ensembles. The vision-language models, even without task-specific fine-tuning, beat the fine-tuned PARSeq and match or exceed several prior published results.

The main limitation is the disconnect between preprocessing evaluation (GROQ placeholder) and recognition evaluation (PARSeq and vision-language models). A fully integrated evaluation preprocessing plus OLMocr-7B on the complete test set would show whether the 91% is achievable with a real recognizer and enable a clean comparison against the SoccerNet 2023 results. That remains the highest-priority next step.

The secondary limitation is PARSeq overfitting, which prevents the ensemble from working as intended. The original design PARSeq as a fast first pass gating expensive VLM inference is sound in principle, and the modular pipeline makes it easy to activate once PARSeq overfitting is resolved. Until then, OLMocr-7B carries the load as primary recognizer.

VI. Future Work

A. Resolving PARSeq Overfitting

This is the most pressing priority, since the intended ensemble design needs a fast, reliable lightweight recognizer. Two directions we have not yet tried look promising. First, synthetic data generation: rendering jersey images programmatically with controlled font variation, lighting, blur, and occlusion would expand the effective training set without requiring more annotated broadcast footage. Models trained on synthetic data that realistically mirrors the target domain often generalize better than those trained on small real datasets, since the synthetic set can cover distribution tails that real data undersamples. Second, progressive fine-tuning: unfreezing layers gradually from the output head toward the early feature extraction layers, rather than fine-tuning all layers at once, gives pretrained representations more room to stay stable while the task-specific parts adapt. This has worked well in other transfer learning settings and has not been tried on PARSeq in this pipeline.

B. Full Evaluation of OVIS2-16B

We only evaluated OVIS2-16B on 17 tracklets. Running inference on the complete test set would show whether 88.24% is representative at scale and enable a fair head-to-head with OLMocr-7B. This requires either hardware with more than 32 GB of VRAM or a quantized version of the model. Both are feasible.

C. Improved Confidence Estimation for Vision-Language Models

The inter-frame consistency heuristic is rough. A better approach would be prompting the model to return a numeric confidence score alongside its prediction, then calibrating those scores against held-out ground truth. Overconfident or underconfident models can be recalibrated via temperature or Platt scaling. Reliable per-frame confidence would bring the aggregation closer to a proper Bayesian combination of evidence.

D. Improved Outlier Elimination with Stronger ReID

The ReID backbone is a ResNet-50 not trained for person re-identification. Replacing it with a model trained on Market-1501 or DukeMTMC would produce more discriminative features and more reliable outlier detection, especially in crowded scenes where the tracker drifts between players. Fig. 10 shows the Swin Transformer architecture [13], which has shown strong ReID results and is a natural upgrade candidate.

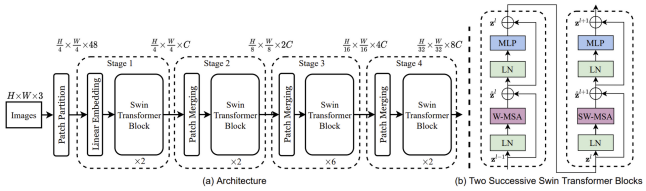

Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

Fig. 10. Swin Transformer architecture (Swin-T). The hierarchical design with shifted window self-attention produces multi-scale feature maps well-suited for person ReID. Replacing the ResNet-50 ReID backbone with a Swin-based model trained on Market-1501 or DukeMTMC is a promising direction for improving outlier elimination quality.

E. Real-Time Lightweight Configuration

The current system with VLM inference is nowhere near real-time. A lightweight version using only NAFNet, YOLOv8, and a well-trained PARSeq should be benchmarked for throughput and latency. This depends on fixing PARSeq overfitting first, but once a reliable lightweight recognizer is available, the modular design makes it easy to strip out the VLM stages.

F. Dataset Expansion and Cross-Competition Generalization

The SoccerNet benchmark covers specific leagues and competitions. Performance here does not guarantee performance on different competitions with different jersey designs, camera setups, and encoding characteristics. Evaluating on additional datasets, or collecting a small eval set from a different competition, would better test generalization. Leagues with unusual fonts or reflective jersey materials are likely to be harder and would expose where the pipeline needs adaptation.

REFERENCES

[1] S. Gerke, K. Muller, and R. Schafer, "Soccer jersey number recognition using convolutional neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, 2015, pp. 17–24.

[2] M. Koshkina and J. H. Elder, "A general framework for jersey number recognition in sports video," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, 2024, pp. 3235–3244.

[3] L. Grad, "Single-stage uncertainty-aware jersey number recognition in soccer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, 2025, pp. 6092–6100.

[4] A. Cioppa et al., "SoccerNet 2023 challenges results," *Sports Engineering*, vol. 27, no. 2, 2024. [Online]. Available: https://arxiv.org/abs/2309.06006

[5] B. Balaji et al., "Jersey number recognition using keyframe identification from low-resolution broadcast videos," in *Proc. ACM MMSports Workshop*, 2023, pp. 123–130.

[6] L. Chen, X. Chu, X. Zhang, and J. Sun, "Simple baselines for image restoration," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2022, pp. 17–33.

[7] A. Abdelhamed, S. Lin, and M. S. Brown, "A high-quality denoising dataset for smartphone cameras," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 1692–1700.

[8] S. Nah et al., "NTIRE 2019 challenge on video deblurring and super-resolution: Dataset and study," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, 2019.

[9] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[10] D. Bautista and H. Atienza, "Scene text recognition with permuted autoregressive sequence models," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2022, pp. 178–196.

[11] H. Lu et al., "OVIS: Open-vocabulary visual instance segmentation," *arXiv preprint arXiv:2407.07281*, 2024.

[12] J. Poznanski et al., "olmOCR: Unlocking trillions of tokens in PDFs with vision language models," *arXiv preprint arXiv:2502.18443*, 2025.

[13] Z. Liu et al., "Swin Transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2021, pp. 10012–10022.