E9 246 – Advanced Image Processing Assignment 2

Bitupan Arandhara
bitupana@iisc.ac.in

Question 1: Graph-Based Image Segmentation using Normalized Cuts

All four test images (*Books*, *Chess*, *Football*, *Zebra*) are converted to grayscale and resized to 100×100 before processing.

Part (a): Original vs. Edge-Aware Affinity

Methodology

The original affinity between pixels i and j combines intensity similarity and spatial proximity:

$$w_{ij}^{\text{orig}} = \exp\left(-\frac{(I_i - I_j)^2}{\sigma_I^2}\right) \exp\left(-\frac{\|X_i - X_j\|^2}{\sigma_X^2}\right).$$

The edge-aware affinity is calculated according to the following formula:

$$w_{ij}^{\text{edge}} = w_{ij}^{\text{orig}} \cdot \exp\left(-\frac{(E_i - E_j)^2}{\sigma_E^2}\right),$$

where E_i is the Sobel gradient magnitude at pixel i, normalised to $[0, 1]$.

The affinity matrices for both cases are computed with parameters $\sigma_I = 0.1$, $\sigma_X = 0.5$, $\sigma_E = 0.01$. The normalised Laplacian $L_{\text{sym}} = D^{-1/2}(D - W)D^{-1/2}$ is computed and its two smallest eigenvectors are extracted. Binary segmentation is obtained by thresholding the Fiedler vector (second-smallest eigenvector) at zero.

Results

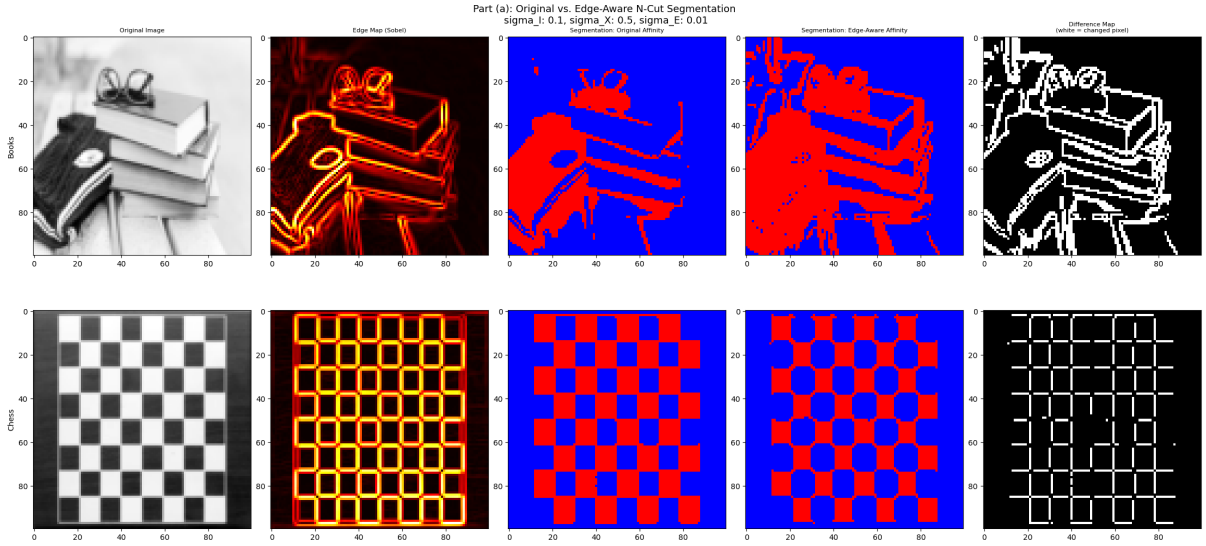


Figure 1: Part (a): Original vs. edge-aware N-Cut segmentation on the *Books* (top) and *Chess* (bottom) images. Columns show: original image, Sobel edge map, segmentation with original affinity, segmentation with edge-aware affinity, and the pixel-level difference map (white = changed label). Parameters: $\sigma_I = 0.1$, $\sigma_X = 0.5$, $\sigma_E = 0.01$.

Observations and Analysis

- **Books image.** The original affinity produces a binary segmentation that splits the image into two regions based primarily on intensity similarity. The edge-aware affinity outputs better the boundaries by penalising connections across strong Sobel gradients. The difference map shows that white pixels are concentrated along the edges of individual books, which confirms that the edge term modifies algorithn to segment at the location of object edges.

- **Chess image.** The segmentation with original affinity correctly captures the alternating black/white tile pattern. Adding the edge-aware term preserves this segmentation but it also sharpens the tile boundaries. The difference map shows very few changed pixels, which indicates that for strongly structured images with sharp edges, the original affinity already performs well.

- **Effect of the edge term.** The exponential factor $\exp(-(E_i - E_j)^2/\sigma_E^2)$ acts in a way such that pixels on opposite sides of a strong gradient receive very low affinity, which discourages the graph cut from crossing object boundaries.

Part (b): Effect of σ_I and σ_X

Methodology

Using both affinity functions, N-Cut segmentation was performed for $\sigma_I \in \{0.05, 0.1, 0.5, 0.8\}$ (with $\sigma_X = 0.5$ fixed) and $\sigma_X \in \{0.25, 0.5, 1.0, 1.5\}$ (with $\sigma_I = 0.1$ fixed) on the four images. σ_E is fixed at 0.01.

Results

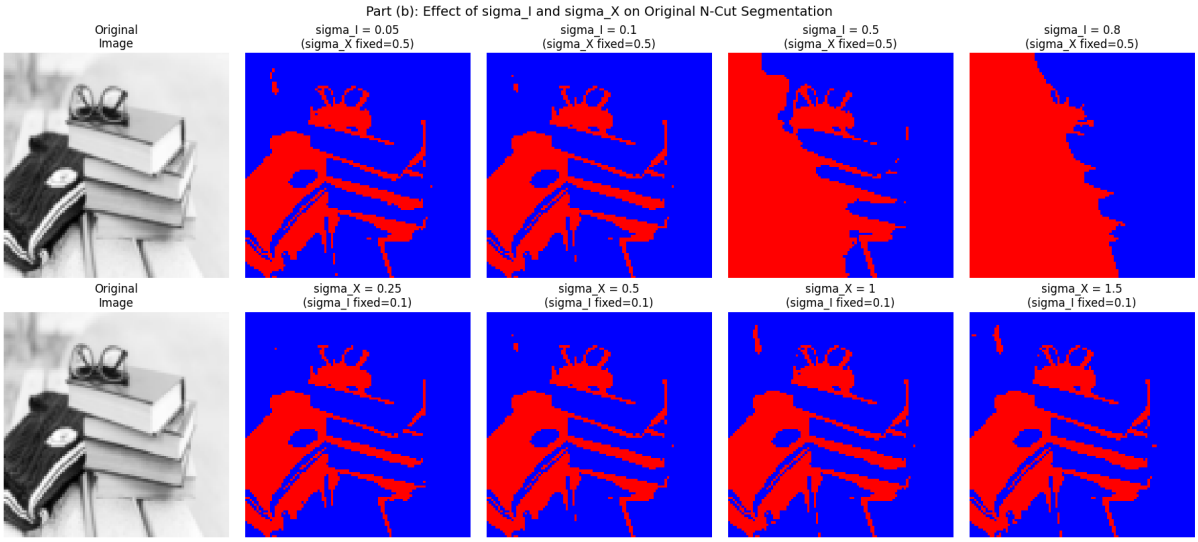


Figure 2: Effect of σ_I (top row) and σ_X (bottom row) on **original** affinity N-Cut segmentation – *Books* image.

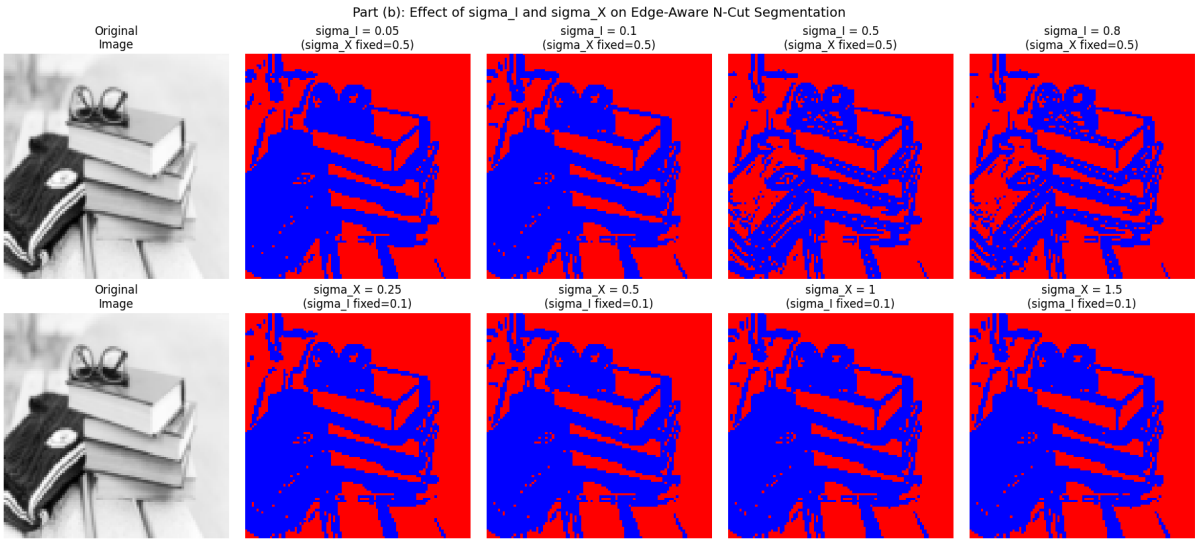
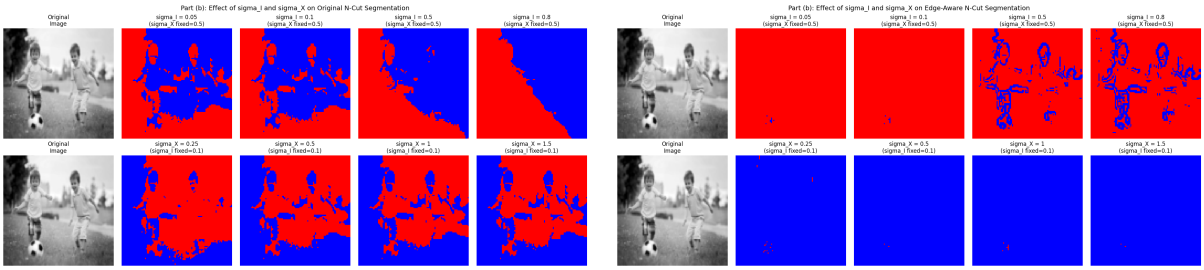
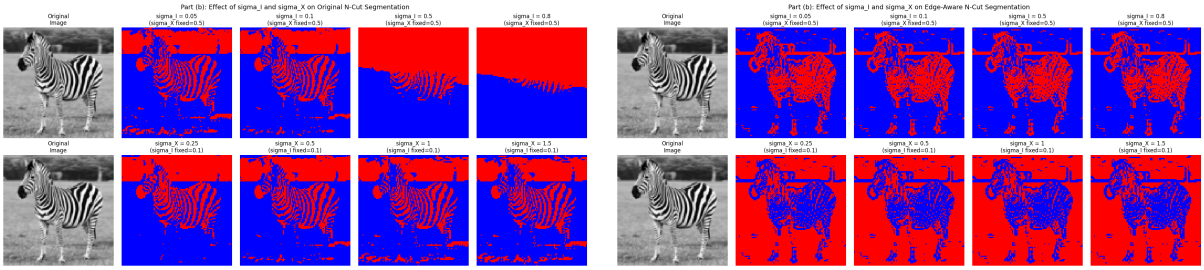


Figure 3: Effect of σ_I (top row) and σ_X (bottom row) on **edge-aware** affinity N-Cut segmentation – *Books* image.



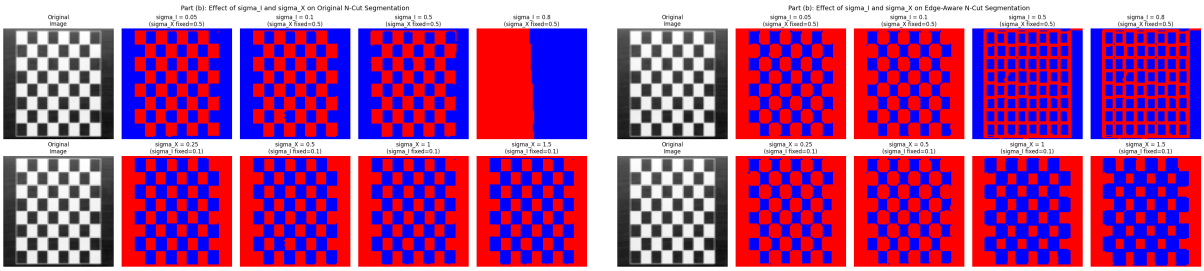
(a) Original affinity (b) Edge-aware affinity

Figure 4: Effect of σ_I (top row) and σ_X (bottom row) on the *Football* image.



(a) Original affinity (b) Edge-aware affinity

Figure 5: Effect of σ_I (top row) and σ_X (bottom row) on the *Zebra* image.



(a) Original affinity (b) Edge-aware affinity

Figure 6: Effect of σ_I (top row) and σ_X (bottom row) on the *Chess* image.

Observations and Analysis

Effect of varying σ_I ($\sigma_X = 0.5$ fixed).

- **Small σ_I ($= 0.05$):** Only pixels with nearly identical intensities receive high affinity. The Fiedler vector separates fine intensity clusters, producing noisy segmentation boundaries. In the *Football* and *Zebra* images, the foreground/background separation is achieved but with some pixel-level noise, which indicates over-segmentation.
- **Moderate σ_I ($= 0.1$):** Performs better than very small σ_I. The *Chess* checkerboard tiles are cleanly segmented; the body of Zebra is well separated from the background; the boys playing football are also distinguishable from the grass.

- **Large σ_I ($= 0.5, 0.8$):** The segmentation degrades to simple geometric splits (e.g., vertical or diagonal segmentation), which indicates *under-segmentation*.

Effect of varying σ_X ($\sigma_I = 0.1$ fixed).

- **Small σ_X ($= 0.25, 0.5$):** As the distance increases affinity decrease rapidly, so only adjacent pixels are strongly connected. The Fiedler vector carries primarily local spatial information. The segmentation is slightly more sensitive to local clusters, but the overall structure remains stable.

- **Large σ_X ($= 1.0, 1.5$):** Pixels far apart influence each other. The Fiedler vector captures a globally consistent partition. The segmentation remains stable across all σ_X values for the *Chess*, *Zebra*, and *Football* images, which indicates that **intensity** is the dominant term when σ_I is already well-tuned.

Interaction with edge-aware affinity: With the edge-aware affinity, the segmentation is more robust to changes in σ_I. Even at $\sigma_I = 0.8$, the edge-aware affinity still respects the primary object boundaries better than the original affinity at the same setting, because the gradient penalty prevents clusters from leaking across strong edges.

Part (c): Multi-Way Segmentation

Strategy 1 – Recursive Two-Way N-Cut

At each iteration the largest region is selected and its exact sub-graph W_{sub} is extracted. The Fiedler vector of W_{sub} is computed via the normalised Laplacian L_{sym}, and the best splitting threshold is found by evaluating $\text{Ncut}(A, B)$ at 30 evenly-spaced thresholds. The split is accepted only if $\text{Ncut}(A, B) < \tau_{\text{ncut}}$ where $\tau_{\text{ncut}} = 1.0$; otherwise the region is left unsplit. Regions smaller than 2% of the image are excluded from splitting. The procedure is repeated until $K = 4$ regions are obtained or until no region could be split further.

Strategy 2 – Simultaneous Partitioning (Multiple Eigenvectors + K-Means)

The $K = 4$ smallest non-trivial eigenvectors of L_{sym} are stacked to form an $N \times K$ embedding matrix Z. K-Means ($K = 4$) is applied to the rows of Z, assigning each pixel to one of K segments.

Results

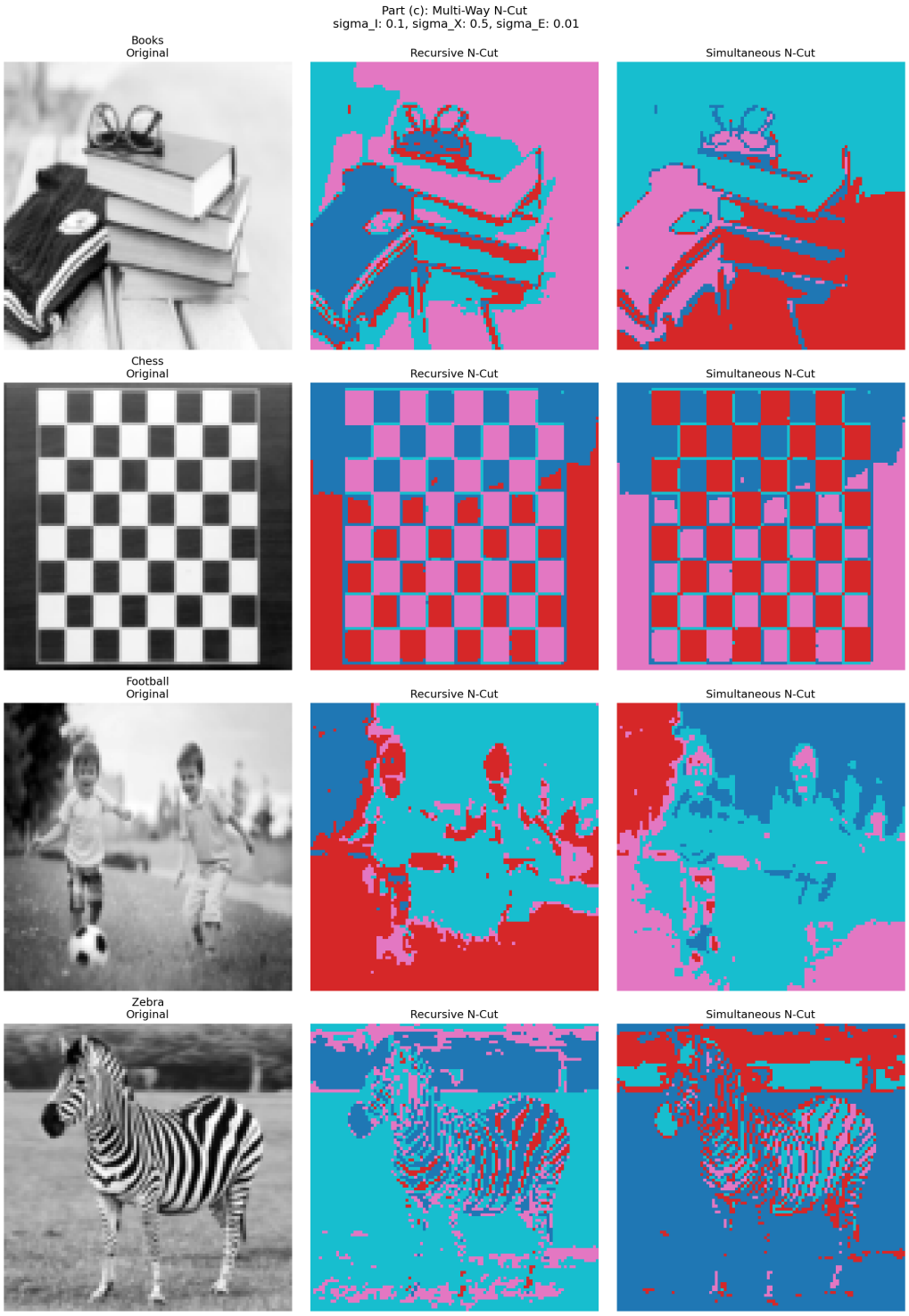


Figure 7: Part (c): Multi-way segmentation ($K = 4$) on four test images. Middle column: recursive two-way N-Cut. Right column: simultaneous N-Cut . Parameters: $\sigma_I = 0.1$, $\sigma_X = 0.5$.

Comparison

The recursive two-way Ncut produces exactly K when τ_{ncut} is permissive and it may produce fewer if all remaining regions fall below the size or Ncut threshold, while the simultaneous N-cut always produces exactly K segments. With $\sigma_I = 0.1$ and $\sigma_X = 0.5$, both methods produce similar meaningful four-region segmentations on all four images (Figure 7), while occasional pixel-level noise is observed in case of the simultaneous case. Both methods struggle with very fine textures (e.g. zebra stripes).

Question 2: Semantic Segmentation with Different Decoder Designs

Dataset and Setup

The dataset consists of **512 soccer-match images** annotated with person instances in the COCO format (394 train, 102 test). The class distribution is highly imbalanced: roughly 1.9% person pixels vs. 98.1% background.

Loss function. Dice Loss is used:

$$\mathcal{L}_{\text{Dice}} = 1 - \frac{2\,\text{TP}}{2\,\text{TP} + \text{FP} + \text{FN}}.$$

Optimizer. AdamW with a different learning rates: 10^{-5} for the encoder, 10^{-3} for the decoder.

Both models are trained for 15 epochs.

Part (a): Single-Stage Upsampling Decoder

Architecture

The encoder is a **ResNet18** backbone pre-trained on ImageNet-1K. The deepest feature map produced by `layer4` has spatial resolution 16×16 and 512 channels (for a 512×512 input). A single transposed convolution upsamples it directly to the full input resolution in one step:

$$\underbrace{16 \times 16 \times 512}_{\text{layer4 output}} \xrightarrow{\text{ConvTranspose2d}(512\to1,\ k=64,\ s=32,\ p=16)} \underbrace{512 \times 512 \times 1}_{\text{segmentation logit}}.$$

Total trainable parameters: **13,273,665**.

Part (b): Progressive Upsampling with Skip Connections

Architecture

The encoder is a modified ResNet18 that stores intermediate feature maps that will be used for skip connections. The decoder progressively upsamples the feature maps using transposed convolution by a factor of 2, with skip features concatenated after the first two upsampling steps:

$$
\begin{aligned}
16\times16\times512 &\xrightarrow{\text{up1: } (512\to256,\, k=3,\, s=2,\, p=1)} 32\times32\times256 \\
&\xrightarrow{\oplus\,\text{skip[layer3]}\ (channels=256)} 32\times32\times512 \\
&\xrightarrow{\text{up2: } (512\to128,\, k=3,\, s=2,\, p=1)} 64\times64\times128 \\
&\xrightarrow{\oplus\,\text{skip[layer2]}\ (channels=128)} 64\times64\times256 \\
&\xrightarrow{\text{up3: } (256\to64,\, k=3,\, s=2)} 128\times128\times64 \xrightarrow{\text{up4}} 256\times256\times64 \xrightarrow{\text{up5}} 512\times512\times1.
\end{aligned}
$$

Each upsampling block (up1–up4) includes `BatchNorm2d` + `ReLU` after the transposed convolution. Skip connections from `layer3` (32×32, 256 ch) and `layer2` (64×64, 128 ch) are concatenated channel-wise before the next upsampling step.

Total trainable parameters: **13,132,417**.

Observations.

- The model reaches **84.92% mIoU by epoch 2**, outperforming the final accuracy of the single-stage decoder (80.66%) in just two epochs. This demonstrates dramatically faster convergence.
- Despite having slightly fewer parameters (13.1M vs. 13.3M), the progressive decoder reaches 91.51% mIoU – a +10.85 point advantage over the single-stage decoder.

Part (c): Quantitative Evaluation and Qualitative Analysis

Quantitative Results

Table 1: Test-set performance comparison of the two decoder designs.

Model	Pixel Accuracy	IoU (person)	IoU (background)	mIoU
Single-stage upsampling	98.87%	53.36%	98.86%	76.11%
Progressive + skip conns.	99.54%	78.75%	99.54%	89.14%
Improvement	+0.67	**+25.39**	+0.68	**+13.03**

The progressive decoder with skip connections outperforms the single-stage baseline on every metric. The highest rate of improvement is in **IoU (person)**: +25.39% (53.36% $\rightarrow$ 78.75%), with **mIoU** increasing from 76.11% in single-stage upsampling to 89.14% in progressive upsampling (13.03% improvement).

Qualitative Analysis

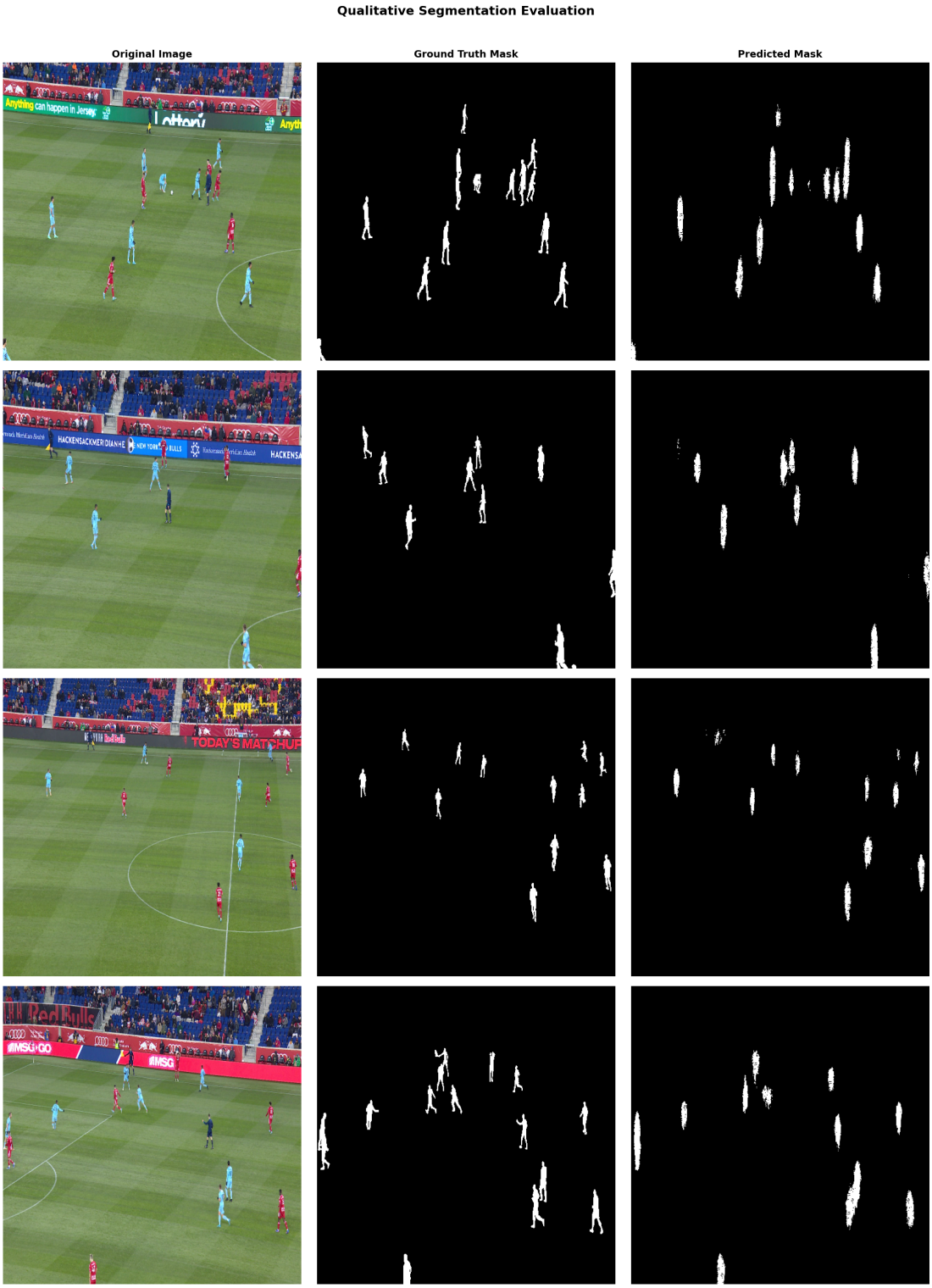


Figure 8: Qualitative analysis of the **single-stage** decoder on 4 test samples. Columns: original image, ground-truth mask, predicted mask.

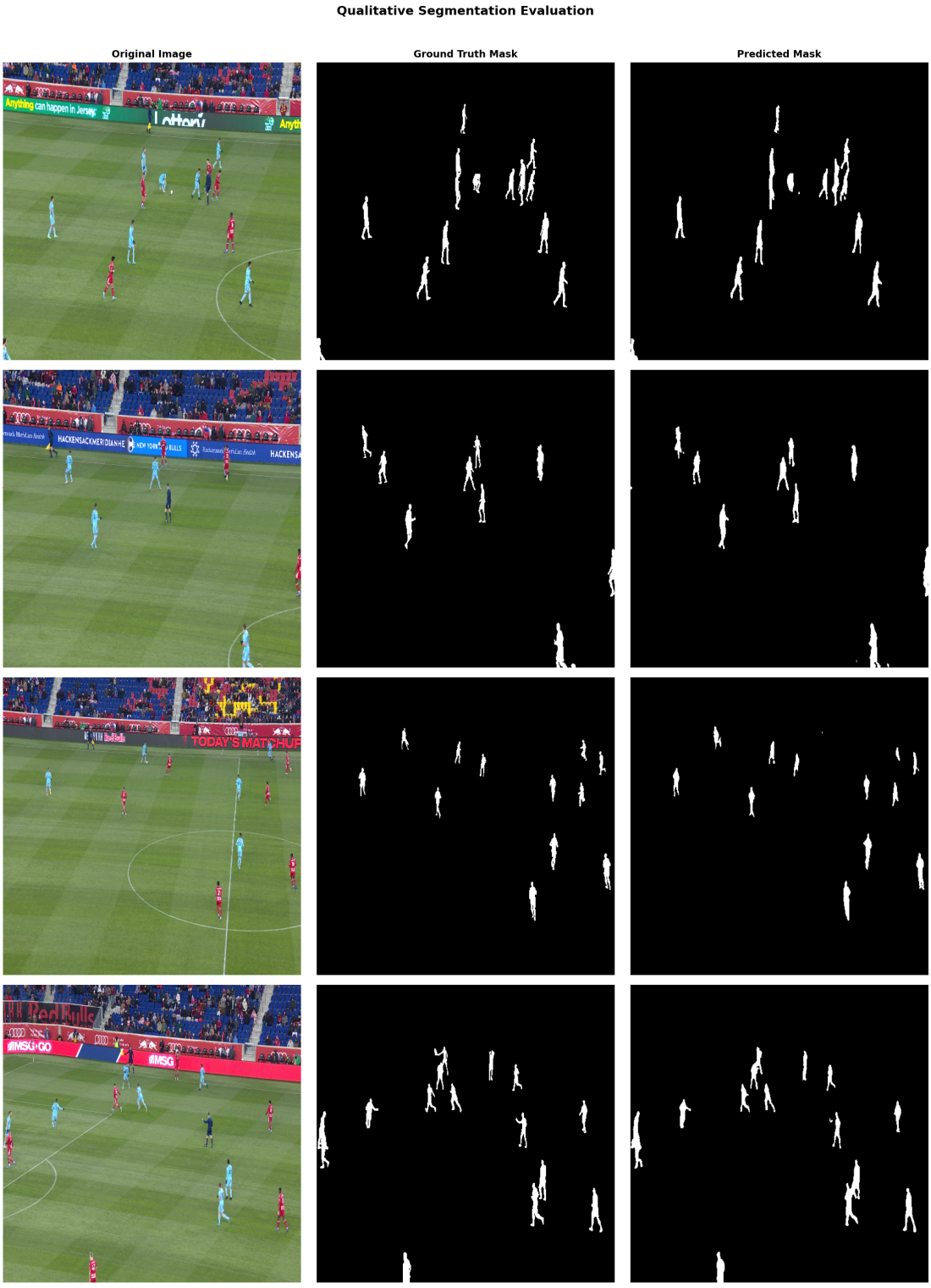


Figure 9: Qualitative analysis of the **progressive** decoder with skip connections on the same 4 test samples. Columns: original image, ground-truth mask, predicted mask.

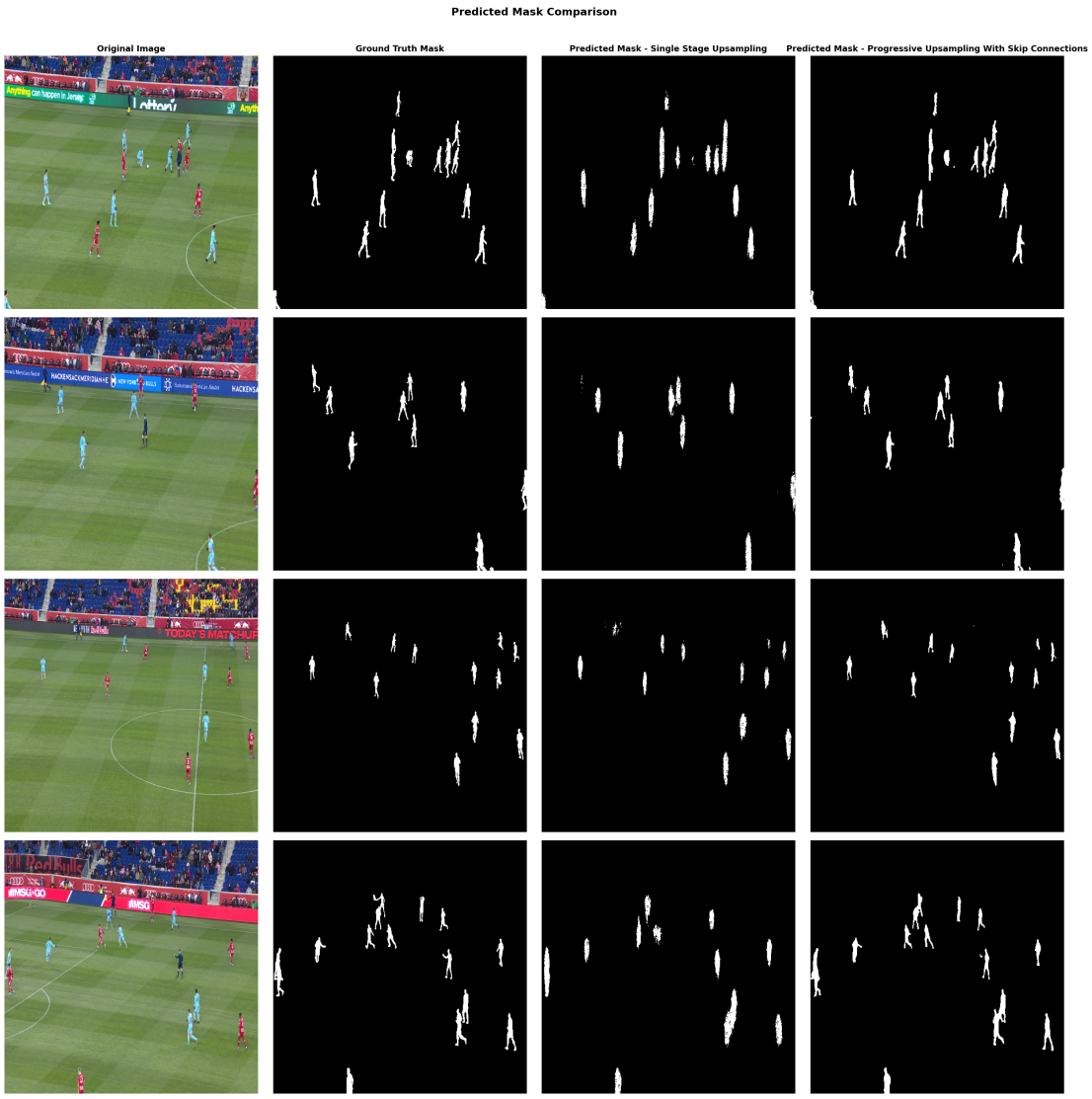


Figure 10: Side-by-side comparison of both models on 4 test samples. Columns: original image, ground-truth mask, single-stage predicted mask, progressive decoder predicted mask.

Role of Progressive Upsampling and Skip Connections

- The **single-stage decoder** produces blob-like person predictions that capture rough player locations but lack fine anatomical detail. The predicted masks are significantly coarser than the ground truth, with rounded boundaries. Smaller players are sometimes missed entirely or merged with nearby players into a single mask.

- The **progressive decoder** produces sharper masks that closely resemble the ground truth. Individual players are better resolved, including smaller figures in the background. The side-by-side comparison in Figure 10 makes this improvement especially striking.

Why the single-stage decoder under-performs: A single ConvTranspose2d with stride 32 must simultaneously learn to upsample by a factor of 32 and produce prediction maps - from a 16×16 feature map that has discarded high-frequency spatial details.

Why progressive upsampling with skip connections succeeds:

1. **Gradual resolution recovery.** Each $2\times$ upsampling step is a easier to learn than a single $32\times$ step.

2. **Skip connections restore spatial detail.** Feature maps from `layer3` (32×32, 256 ch) and `layer2` (64×64, 128 ch) carry high-resolution edge and texture information. Concatenating these maps provides the decoder with precise spatial information for boundary localization.