Appendices

Contents

1. Introduction **1**

2. Related Work **2**

3. Method **2**
3.1. Overview . . . 2
3.2. Training Data Curation . . . 3
3.3. Architect: Spatial Layout Generation . . . 3
3.3.1 . Architect-A: Bounding-Box Co-ord. Regression . . . 3
3.3.2 . Architect-B: T2I-based Layout Generation . . . 4
3.4. Artist: Identity-Preserving Rendering . . . 4
3.4.1 . GRPO Training with Compositional Rewards . . . 5
3.5. Dropping and Sharing Tokens . . . 6
3.6. Training Curriculum . . . 6

4. Experiments **6**
4.1. Experimental Setup . . . 6
4.2. Results . . . 6

5. Conclusion **8**

A Introduction **13**

B Implementation Details and Hyperparameters **13**
B.1. Training Dataset . . . 13
B.2. Models . . . 14
B.3. Training . . . 14
B.4. Inference . . . 16

C Extended Method Section **17**
C.1. Canvas Construction Pipeline(Training) . . . 17
C.2. Frontal Pose Reward Formulation . . . 17
C.3. Canvas Layout Construction(Inference) . . . 18
C.4. Body-Pose Controlled Artist . . . 19
C.5. Training Algorithms . . . 19
C.5.1. Architect-A Training . . . 19
C.5.2. Architect-B Training . . . 20
C.5.3. Hungarian Centroid Matching for Face Rewards . . . 20
C.5.4. Artist Training . . . 20

D Quantitative Results **22**
D.1. Data Sorting for Coordinates Regression . . . 22
D.2. Standalone Architect Performance . . . 22
D.3. Grid Search on Reward Weights . . . 22
D.4. Performance across varying number of people . . . 24
D.5. Latency Analysis and Speed Comparisons . . . 24
D.6. Unified Architecture Variant (Architect-C) . . . 24

D.7. Off-the-Shelf Layout Generation as Architect . . . 25

E. Qualitative Results **25**
E.1. Qualitative Comparison on Multi-ID Test . . . 26
E.2. Pose-Controlled Artist Results . . . 26
E.3. Effectiveness of Token Sharing . . . 27
E.4. Naive Face Matching vs. Hungarian Centroid Face Matching . . . 28
E.5. Architect Variant Comparison . . . 28
E.6. Visualizing Frontal Pose Scores . . . 30
E.7. HPSv3 Reward Impact . . . 30
E.8. Generalization to Multi-Object and Multi-Human+Object Scenes . . . 30

F. Limitations and Future Work **32**

G Disclosure of LLM Use **32**

A. Introduction

This appendix is comprehensive supplementary material to support our main paper. We organized this content into six sections that progressively detail our implementation, methodology, experimental results, and analysis.

Section B presents complete implementation details and hyperparameters for all components of Ar2Can, including both the Architect variants and our Artist module. This includes the final run training curves. Section C extends our method description with detailed algorithms, architectural specifications, and training procedures omitted from the main paper due to space constraints. Section D provides extensive quantitative analysis. This includes any additional ablation studies, scaling experiments, and component-wide performance breakdowns. Section E presents comprehensive qualitative results, visualizing failure modes, architectural comparisons, and the effectiveness of our proposed components. Finally, Section F discusses current limitations of our approach and outlines promising directions for future research. Together, these sections provide the complete technical details necessary for reproducing our results and understanding the full scope of our contributions.

B. Implementation Details and Hyperparameters

This section provides comprehensive implementation details for reproducing our results. We begin with our training dataset construction and prompt curation strategy (Section B), followed by model architectures and detection tools used throughout our pipeline. We then detail the complete training procedures for both Architect variants and the Artist, including hyperparameters, optimization schedules, and hardware configurations. Finally, we describe our inference setup and computational requirements.

B.1. Training Dataset

Our training dataset consists of 60,000 carefully curated prompts designed to capture diverse multi-human scenarios. Each prompt describes group scenes containing 2-7 people engaged in various activities and contexts. The captions were generated using GPT-5 to ensure high-quality, diverse descriptions that encompass a wide range of:

- **Social contexts**: Family gatherings, business meetings, friend groups, professional teams, recreational activities
- **Settings**: Indoor and outdoor environments, formal and informal occasions, workplace and leisure contexts
- **Activities**: Collaborative tasks, social interactions, professional activities, recreational pursuits
- **Group compositions**: Varying numbers of individuals (2-7) with diverse demographic representations

for 30% of the prompts, we add the tag Everyone is looking at the camera. These prompts activate the frontal pose reward during training. The prompts were designed to avoid overlap with evaluation datasets while maintaining sufficient diversity to train robust multi-human generation capabilities. The following are 5 examples of these prompts.

- Seven people on the desert dunes, hazy sun, diverse faces, clear faces visible, studio-quality, vivid detail

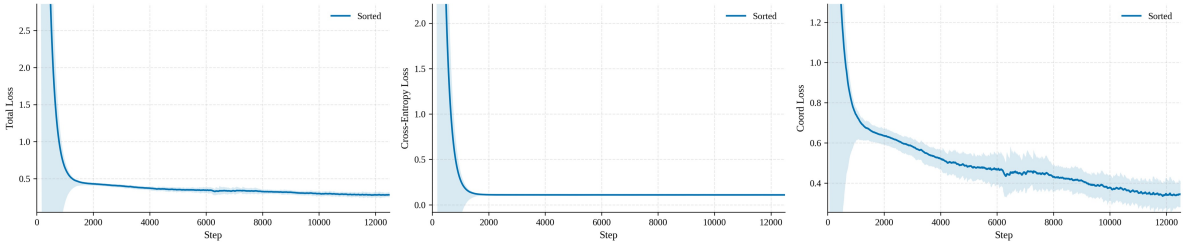
- Six people in an astronomy studio, Clean composition, Professional portrait, Portrait photography, Soft shadows, Natural lighting, Even exposure
- Three people in an aviation observatory, Sharp focus, Clean composition, Bokeh background, Color graded, Smiling expressions, Well lit
- Five people in a dawn-lit bakeshop, Studio quality, Even exposure, Group harmony, Cinematic lighting, Portrait photography, Soft shadows
- Seven people on a coastal boardwalk, afternoon light, diverse faces, clear faces visible, ultra-realistic, 8K resolution. Everyone is looking at the camera.

The canvas is then constructed for each prompt as discussed in Algorithm 1, with the sampling probabilities $p_1 = 0.5, p_2 = 0.4, p_3 = 0.1$. We try to reduce the amount of synthetic faces, even though DisCo has been trained to generate diverse faces.

B.2. Models

Architect-A uses Qwen-2.5-0.5B. **Architect-B** uses Flux-Schnell (1B parameters, 4 steps). The **Artist** is a Flux-Kontext (12B parameters). We use RetinaFace [11] (ResNet-50 backbone) for face detection with confidence threshold 0.5. To compute ArcFace embeddings [10], we use the Anetlopev2[1] model, producing 512-dimensional features. For pose computation, we use the rtmlib library [2] with their default model.

B.3. Training



a) Total Loss b) Cross-Entropy Loss c) Co-Ordinate Loss

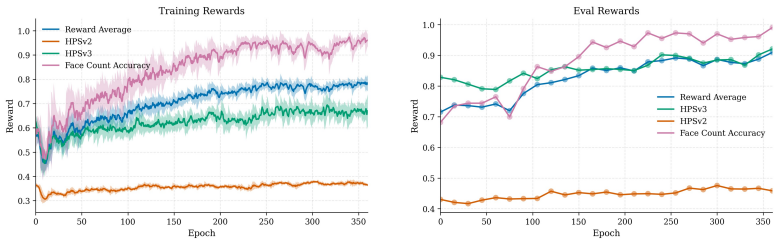
Figure B.1. **Training curves of Architect-A (LLM-based).** The cross-entropy loss converges quickly, indicating that the model stabilizes its estimation of the number of bounding boxes after roughly 2000 steps. After this point, training is dominated by the optimization of bounding-box coordinate regression.

For **Architect-A**, we fine-tune Qwen for 20K steps using AdamW with a learning rate of 10^{-5} and a batch size of 128. We begin with a 100-step warm-up phase in which the learning rate increases linearly from zero to its maximum value, followed by a cosine decay schedule that gradually reduces it back to zero. We set $\lambda_{\text{coord}} = 0.8$ to balance the coordinate regression loss and the cross-entropy loss. All coordinates are normalized to the range $[0, 1]$. The coordinate regression head f_{value} is a 4-layer MLP. For coordinate embedding f_{embed}, we first map the normalized coordinates back to the original $[0, 1024]$ range, convert them to integer values, and feed them into a discrete embedding layer. These embeddings are added to the token embeddings to provide positional information and allow the model to distinguish different coordinate tokens. The streams of f_{value} and f_{embed} are activated only when f_{token} predicts that the next token corresponds to a coordinate token <C>. Architect-A training takes approximately 6 GPU-hours on a single A100-80GB.

Figure B.1 presents the SFT training progression of Architecture-A. The model adapts rapidly to the counting task, with the cross-entropy loss converging within the first 2000 steps. After this point, the remaining optimization primarily focuses on refining bounding-box coordinate regression. Although the bounding-box count is an important component of the task, the learning of coordinate distributions plays a central role in shaping the model's spatial understanding. The convergence behavior of the coordinate loss reflects how effectively the model learns the geometric patterns in the data.

Architect-B Training. We implement Architect-B using the public `flow_grpo`[3] framework with Flux-Schnell pipeline, training in bf16 mixed precision on 512×512 images. To save on training memory, we train a LoRA with Rank=64 instead

[1] https://github.com/deepinsight/insightface
[2] https://github.com/Tau-J/rtmlib
[3] https://github.com/yifan123/flow_grpo

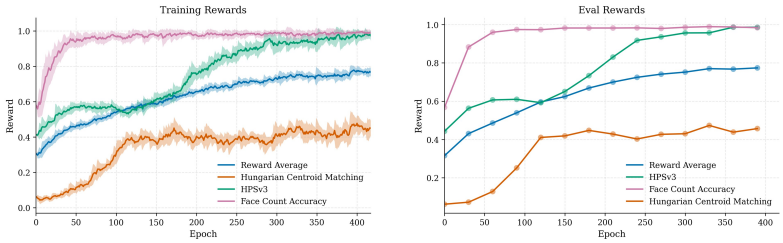


a) Arch-B Training Rewards b) Arch-B Validation Rewards(512x512)

Figure B.2. **Architect-B training and validation reward curves.** We observe steady improvement in both count accuracy and prompt alignment/aesthetic quality (HPS) rewards during training and evaluation, with count accuracy stabilizing earlier than HPS scores.

of training the full network. Training uses 3 timesteps for reward computation and evaluation. We train for 240 epochs with batch sizes of 3 (train) and 16 (test), with a group size of 21. The compositional reward function combines count accuracy ($\alpha = 0.5$) and prompt alignment via HPS. We use both HPSv2 and HPSv3 ($\beta = 0.25$ each), with KL regularization weight $\beta_{\text{KL}} = 0.01$ to stabilize learning. Training is distributed across 8×H100-80GB GPUs on a single node, of which 1 dedicated GPU is used for HPSv3 reward computation and 7 GPUs for training. We use a learning rate of 1×10^{-4} with EMA enabled. Face detection for count accuracy uses blob analysis on the 3-step generated images. Total training time to 240 epochs is approximately **140 GPU-hours**.

Figure B.2 demonstrates the training progression of Architect-B across both reward components throughout the GRPO fine-tuning process. The curves show consistent improvement in count accuracy and prompt alignment (HPSv3) metrics during both training and evaluation phases. Count accuracy shows rapid improvement in the first 200 epochs, stabilizing around epoch 240, similar to HPSv3 scores. The model achieves strong performance on the test set by epoch 240, with minimal overfitting observed. Please note that these scores are on the val set(different from our testsets in the paper), and also for model inference at 512x512, at 18 timestep inference; hence, the numbers are different from scores in the paper.



a) Artist Training Rewards b) Artist Validation Rewards(512x512)

Figure B.3. **Artist training and evaluation reward curves.** We observe steady improvement in all compositional rewards (count accuracy, HPSv3, face matching, and pose alignment) during training and evaluation. The curriculum transition at epoch 100 is visible as a brief perturbation in HPSv3, after which training stabilizes with continued improvement.

Artist Training. We implement the Artist using the public `flow_grpo` framework with Flux-Kontext pipeline, training in bf16 mixed precision on 512×512 images. To save on training memory, we train a LoRA with Rank=64 instead of training the full network. Training uses 18 timesteps for reward computation and evaluation, with classifier-free guidance of 2. We train for 300 epochs with batch sizes of 3 (train) and 16 (test), with a group size of 21. The compositional reward function combines count accuracy ($\alpha = 0.2$), image quality via HPSv3 ($\beta = 0.4$), hungarian centroid face matching ($\zeta = 0.3$), and frontal pose alignment ($\eta = 0.1$) components, with KL regularization weight $\beta_{\text{KL}} = 0.00$ to stabilize learning. We apply curriculum learning with transition epoch $\tau = 100$, sampling equally from 2-3 person scenes for the first 100 epochs, then uniformly from 2-7 person scenes. Training is distributed across 24 H100-80GB GPUs on 3 nodes, consisting of 3 dedicated GPUs for HPSv3 reward servers (1 per node). Hence, we use 21 GPUs for training (7 per node) and 3 for the reward servers. We use a learning rate of 1×10^{-4} with EMA enabled. Total training time to 300 epochs is approximately **1000 GPU-hours**, and on our setup it takes **2 days**.

Algorithm 1 Canvas Construction Pipeline

Require: Data sources: multi-view $\mathcal{D}_1$, single-view $\mathcal{D}_2$, synthetic $\mathcal{D}_3$ with probabilities p_1, p_2, p_3
Require: Pose mode flag: `use_pose`
Ensure: Training sample: canvas C, reference faces $\{I_{\text{ref},i}\}$, target image I_{DisCo}

Stage 1: Scene Generation
Sample target count $n \sim \text{Uniform}(2, 7)$
Generate $I_{\text{DisCo}} \sim \text{Flux-DisCo}(n)$ ▷ Multi-person scene
Stage 2: Face Localization
Detect face bounding boxes $\{b_1^{\text{DisCo}}, \ldots, b_n^{\text{DisCo}}\} \leftarrow \text{RetinaFace}(I_{\text{DisCo}})$
if `use_pose` **then**
    Estimate human poses $\{p_1^{\text{DisCo}}, \ldots, p_n^{\text{DisCo}}\} \leftarrow \text{PoseEstimator}(I_{\text{DisCo}})$
end if
Stage 3: Reference Selection & Augmentation
for each face location $i = 1, \ldots, n$ **do**
    Sample source $k \sim \text{Categorical}(p_1, p_2, p_3)$
    Sample reference face $I_{\text{ref},i}$ from $\mathcal{D}_k$
    if $k = 1$ **then** ▷ Multi-view
        Use secondary view directly
    else if $k = 2$ or $k = 3$ **then** ▷ Single-view or synthetic
        Generate secondary view via:
            PuLID [17] (pose/expression variation), or
            Augmentations (rotation $\pm 15°$, flip, brightness)
    end if
end for
Stage 4: Canvas Composition
Initialize blank canvas C with resolution matching I_{DisCo}
for each face $i = 1, \ldots, n$ **do**
    Paste $I_{\text{ref},i}$ onto C at location b_i^{DisCo}
end for
if `use_pose` **then**
    for each person $i = 1, \ldots, n$ **do**
        Overlay pose skeleton p_i^{DisCo} onto C
    end for
end if
return C, $\{I_{\text{ref},i}\}_{i=1}^{n}$, I_{DisCo}

Figure B.3 demonstrates the training progression of the Artist across all four compositional reward components throughout the GRPO optimization process. The curves show consistent improvement in count accuracy, image quality (HPSv3), spatially-grounded face matching(including frontal pose alignment) during both training and evaluation phases. Face matching rewards exhibit the most dramatic improvement, particularly in the first 150 epochs, reflecting the model's learning to preserve identities while maintaining spatial correspondence. Count accuracy stabilize relatively quickly, while HPSv3 continues gradual improvement throughout training. The curriculum transition at epoch 100 (from 2-3 person scenes to all counts) is visible as a brief perturbation in the HPSv3 curve, after which training continues smoothly. Please note that the evaluation scores are on the val set(different from our testsets in the paper), and also for model inference at 512x512, at 18 timestep inference; hence, the numbers are different from scores in the paper.

B.4. Inference

At inference, we sample an Architect layout and generate the final image. For Architect-A, layout generation is deterministic (greedy decoding). For Architect-B, we use 4-step flow sampling with 0 guidance scale. The Artist uses 28-step generation with guidance scale of 2. For the Artist, we apply the token saving and pass input faces through separate canvases. In overlapping tokens, we pass the same RoPE positional ID but pass both the tokens. All evaluations are at 1024×1024 resolution.

Baselines. We evaluate all baseline methods using their official open-source implementations with default parameters unless otherwise noted. For all the baseline methods, evaluations are at 1024×1024 resolution.

UMO-UNO [7]: We obtained the code from the official repository[4] and used the default configuration with 25 inference steps and default guidance scale as recommended by the authors.

X-Verse [6]: We obtained the code from the official repository[5] and followed the default settings with 28 inference steps and default guidance scale.

UMO-OmniGen2 [7]: We obtained the code from the official repository[6] and used the recommended configuration with 50 inference steps and default guidance scale.

Dream-O [30]: We obtained the code from the official repository[7] and adopted the recommended Flux.1-turbo variant with 12 inference steps and default guidance scale as specified in the documentation.

ID-Patch [53]: We obtained the code from the official repository[8] and used all default parameters from the official implementation without modification.

WithAnyone [48]: We obtained the code from the official repository[9] and used the default parameters except for the SigLIP weight, which we set to 0.5 as we found it to be the optimal trade-off between identity preservation and prompt alignment.

C. Extended Method Section

This section expands upon the methodological components described in the main paper with additional algorithmic details and formulations. We provide the complete canvas construction pipeline (Algorithm 1) that generates our hybrid training data, followed by detailed formulations for our frontal pose reward and our optional body pose controlled Artist variant. We then present step-by-step training algorithms for all three components: Architect-A (supervised fine-tuning), Architect-B (GRPO with count and quality rewards), and the Artist (GRPO with compositional rewards including our novel Hungarian centroid matching). These algorithms complement the high-level descriptions in the main paper and provide the precise implementation details needed for replication.

C.1. Canvas Construction Pipeline(Training)

We construct each training sample via a four-stage pipeline (as expanded in Algorithm 1). This produces hybrid samples combining synthetic multi-person scenes with real reference faces. Figure C.1 illustrates this process with example canvas samples showing reference faces drawn from different data sources ($\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_3$).

We start the process by generating a DisCo [4] scene with N people and detecting face locations (Stage 1-2). For generating the canvas in a pose-controlled training, we additionally estimate human pose skeletons from the DisCo image. Then, we initialize a blank canvas and replace each synthetic face with a reference face from our curated sources (Stage 3-4). In pose mode, we overlay the estimated pose skeletons onto the canvas, providing additional spatial guidance for action and pose alignment.

Each training sample comprises three components: the constructed canvas (with reference faces and optionally pose overlays), the original reference face images, and the DisCo generated target image. As illustrated in Figure C.1, reference faces are sampled from different data sources. These are $\mathcal{D}_1$ (multi-view), $\mathcal{D}_2$ (single-view with augmentation), and $\mathcal{D}_3$ (synthetic). This enables diverse identity representation while maintaining authentic facial appearances.

An important consideration is that while we show a single canvas with all faces in it, the actual training sample has multiple canvases, each with a separate face. This is so that we can provide the shared RoPE encodings illustrated in 5.

C.2. Frontal Pose Reward Formulation

We design a lightweight 2D frontal pose reward using 5-point facial landmarks (left eye, right eye, nose, left mouth, right mouth) detected by RetinaFace [11]. For each detected face i, we compute a frontality score δ_i combining roll angle and yaw asymmetry:

Roll Angle: We compute the roll angle from eye landmarks: $\theta_{\text{roll},i} = |\arctan\left(\frac{y_{\text{re}} - y_{\text{le}}}{x_{\text{re}} - x_{\text{le}}}\right)|$, where $(x_{\text{le}}, y_{\text{le}})$ and $(x_{\text{re}}, y_{\text{re}})$ are left and right eye coordinates.

[4] https://github.com/bytedance/UMO
[5] https://github.com/bytedance/XVerse
[6] https://github.com/VectorSpaceLab/OmniGen
[7] https://github.com/bytedance/DreamO
[8] https://github.com/bytedance/ID-Patch
[9] https://github.com/Doby-Xu/WithAnyone

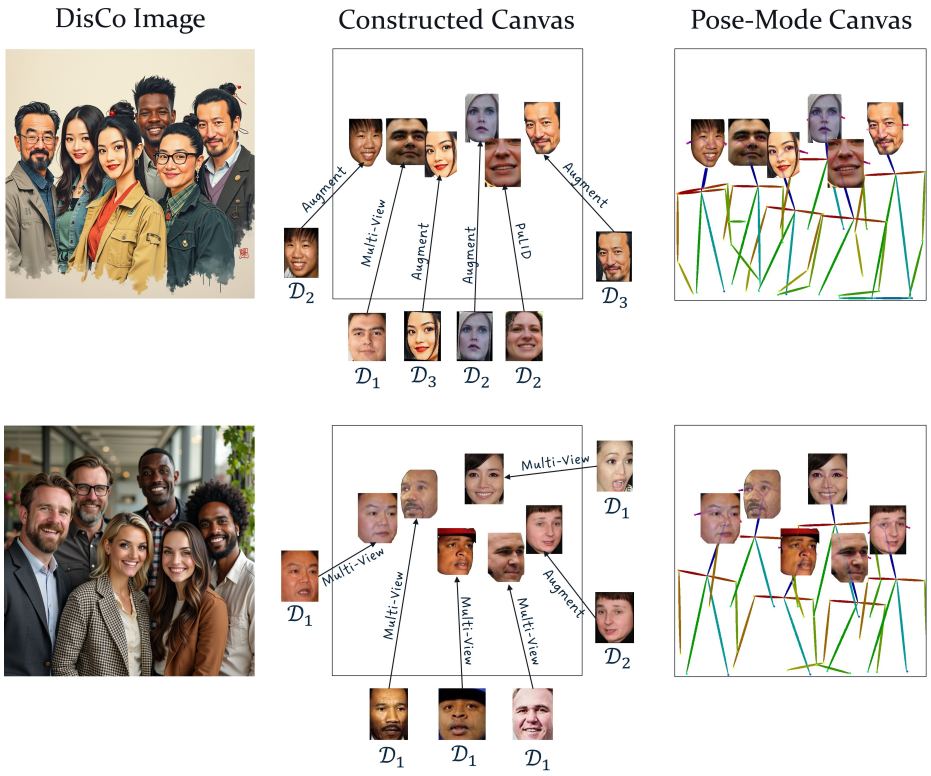


Figure C.1. Illustration of the Canvas Construction Pipeline. **Left**: Image generated by DisCo [4], using the input prompt. **Middle**: Standard canvas. Each canvas combines reference faces sampled from different data sources: $\mathcal{D}_1$ (multi-view references with secondary views), $\mathcal{D}_2$ (single-view references with PuLID-generated or augmented secondary views), and $\mathcal{D}_3$ (the original synthetic face from DisCo). **Right**: If we want to perform pose-mode training, we overlay the human pose skeletons estimated from the DisCo images. We make sure to blend with lesser intensity in face bbox regions.

Yaw Asymmetry: We compute left-right facial asymmetry as $\Delta_{\text{sym},i} = |\|\mathbf{p}_{\text{le}} - \mathbf{p}_{\text{nose}}\| - \|\mathbf{p}_{\text{re}} - \mathbf{p}_{\text{nose}}\||$, where $\|\mathbf{p}_a - \mathbf{p}_b\|$ denotes Euclidean distance between landmarks.

The frontality score applies Gaussian penalties to both components:

$$\delta_i = \exp\left(-\frac{\theta_{\text{roll},i}^2}{2\sigma_{\text{roll}}^2}\right) \cdot \exp\left(-\frac{\Delta_{\text{sym},i}^2}{2\sigma_{\text{yaw}}^2}\right) \tag{7}$$

where $\sigma_{\text{roll}} = 15.0$ degrees and $\sigma_{\text{yaw}} = 0.25$ (normalized by inter-ocular distance). We threshold δ_i to zero if $\delta_i < 0.9$ to create a sharp distinction between frontal and non-frontal faces. The frontal pose reward is: $r_{\text{pose}}(x, \mathcal{L}) = \frac{1}{O}\sum_{i=1}^{O} \delta_i$, where O is the number of detected faces. This reward is activated when prompts contain "Everyone is looking at the camera" (30% of training data). We empirically observe this formulation to be lightweight and effective for reducing copy-paste artifacts in our multi-human generation setting.

C.3. Canvas Layout Construction(Inference)

At inference, the canvas layout $\mathcal{L}$ is constructed from the Architect-predicted bounding boxes and the user-provided reference images. For each reference image, we first apply a segmentation model to extract the face region, removing background elements before placement. The segmented face is then aligned to its corresponding Architect-predicted bounding box coordinates using a face detection model, which localizes facial landmarks to ensure accurate spatial correspondence between the reference face and its target location on the canvas. Each aligned, segmented face is then pasted onto a blank white canvas at the predicted box location, producing the final layout $\mathcal{L}$ that is fed to the Artist alongside the text prompt and reference images.

C.4. Body-Pose Controlled Artist

Note: This section describes an optional pose-controlled Artist variant explored in our experiments (Section E). The main Ar2Can model uses only the frontal face reward described above.

Our pose-controlled Artist variant conditions on full-body pose skeletons overlaid on the canvas (pose-mode in Figure C.1), providing explicit control over body poses and actions beyond just face locations.

We introduce a body pose reward based on keypoint matching. For each generated image, we detect human keypoints in both the reference pose layout and generated image using a pose estimation model (rtmlib). Next, we compute **Object Keypoint Similarity (OKS)**[10] between each pair of reference and generated persons:

$$\text{OKS}_j = \frac{1}{|K_j|} \sum_{k \in K_j} \exp\left(-\frac{d_k^2}{2a_j\kappa_k^2}\right) \tag{8}$$

where d_k is the Euclidean distance between matched keypoints (e.g., left elbow in reference vs. generated), a_j normalizes by the person's area, κ_k is a per-keypoint constant from the COCO evaluation protocol (e.g., $\kappa_k = 0.026$ for shoulders, 0.107 for eyes), and K_j contains number of visible keypoints.

Since the number of detected people may differ from the reference count, we use **Hungarian matching** to find the optimal one-to-one correspondence between reference and generated persons that maximizes total OKS. The body pose reward aggregates across matched pairs:

$$r_{\text{body-pose}}(x, \mathcal{L}) = \frac{1}{O} \sum_{j=1}^{O} \text{OKS}_j \tag{9}$$

where O is the number of matched pairs. This reward encourages the model to reproduce both overall scene layout and fine-grained limb positions.

Training: When training the pose-controlled Artist, this body pose reward replaces the frontal face reward. The pose-controlled variant achieves accurate pose alignment but exhibits increased copy-paste artifacts compared to the standard Artist (Figure E.2), representing a trade-off between pose controllability and rendering naturalness.

C.5. Training Algorithms

This section provides concise training algorithms for each component of Ar2Can. Algorithm 2 describes the supervised fine-tuning procedure for Architect-A, Algorithm 3 details the GRPO-based training for Architect-B, Algorithm 4 presents our Hungarian centroid matching procedure for spatially-grounded face matching, and Algorithm 5 presents the complete Artist training procedure with compositional rewards.

Algorithm 2 Architect-A Training (Supervised Fine-tuning)

Require: Pre-trained LLM θ_{LLM} (Qwen-2.5-0.5B), dataset $\mathcal{D} = \{(p_i, \mathcal{L}_i)\}$
1: Initialize heads $f_{\text{value}}, f_{\text{embed}}$; extend tokenizer with <SoL>, <EoL>, <C>
2: **for** each batch $\{(p_j, \mathcal{L}_j)\}$ in $\mathcal{D}$ **do**
3: Sort coordinates in $\mathcal{L}_j$ (left-to-right, top-to-bottom)
4: Forward: $h = \theta_{\text{LLM}}(p_j)$; predict tokens $\hat{y} = f_{\text{token}}(h)$; coords $\hat{b} = f_{\text{value}}(h)$
5: Compute: $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}}(\hat{y}, y_j) + \lambda_{\text{coord}}[\mathcal{L}_{\text{gIoU}}(\hat{b}, b) + \|\hat{b} - b\|_1]$
6: Update: $\theta_{\text{LLM}}, f_{\text{value}}, f_{\text{embed}} \leftarrow \text{Adam}(\nabla\mathcal{L}_{\text{total}})$
7: **end for**
8: **return** Fine-tuned Architect-A

C.5.1. Architect-A Training

Algorithm 2 presents the supervised fine-tuning procedure for our LLM-based Architect-A. As describd in Section 3.3.1 of the main text, the model is trained to autoregressively generate structured layouts with special tokens, while coordinate regression and embedding heads enable continuous bounding box prediction.

[10] https://cocodataset.org/#keypoints-eval

Algorithm 3 Architect-B Training (GRPO)

Require: Pre-trained T2I π_{ref} (Flux-Schnell), prompts $\mathcal{P}$, group size M

Initialize policy $\pi_\theta \leftarrow \pi_{\text{ref}}$
for each prompt p in $\mathcal{P}$ **do**
  Sample group: $\{x_1, \ldots, x_M\} \sim \pi_\theta(\cdot|p)$
  Compute rewards: $r_i = \alpha \cdot \mathbb{1}[n_{\text{pred}}(x_i) = n_{\text{target}}] + \beta \cdot \text{HPSv3}(x_i, p)$
  Compute advantages: $A_i = (r_i - \mu_G)/(\sigma_G + \epsilon)$
  GRPO loss: $\mathcal{L} = \sum_{i=1}^{M} A_i \log \frac{\pi_\theta(x_i|p)}{\pi_{\text{ref}}(x_i|p)} - \beta_{\text{KL}}\text{KL}(\pi_\theta||\pi_{\text{ref}})$
  Update: $\theta \leftarrow \text{Adam}(\nabla\mathcal{L})$
end for
return Fine-tuned Architect-B

C.5.2. Architect-B Training

Algorithm 3 describes the GRPO procedure for Architect-B, fine-tuning the model to generate accurate person counts and plausible spatial layouts. This is an extension from the explanation in Section 3.3.2 of the main text.

Algorithm 4 Hungarian Centroid Matching

Require: Image x, layout $\mathcal{L} = \{b_i^{\text{pred}}\}$ with centroids $\{c_i^{\text{pred}}\}$, refs $\{I_{\text{ref},i}\}$

Detect faces: $\{b_j^{\text{det}}\} \leftarrow \text{RetinaFace}(x)$; compute centroids $\{c_j^{\text{det}}\}$
Build cost matrix: $C_{ij} = \|c_i^{\text{pred}} - c_j^{\text{det}}\|_2$
Hungarian assignment: $\pi^* = \arg\min_\pi \sum_{i=1}^{\min(N,O)} C_{i,\pi(i)}$
for $i = 1$ to N **do**
  if i has valid match **then**
    $e_i^{\text{ref}} = f_{\text{ArcFace}}(I_{\text{ref},i})$; $e_{\pi^*(i)}^{\text{gen}} = f_{\text{ArcFace}}(\text{crop}(x, b_{\pi^*(i)}^{\text{det}}))$
    $s_i = \frac{e_i^{\text{ref}} \cdot e_{\pi^*(i)}^{\text{gen}}}{\|e_i^{\text{ref}}\| \|e_{\pi^*(i)}^{\text{gen}}\|}$
  else
    $s_i = 0$
  end if
end for
return $r_{\text{face}} = \frac{1}{N} \sum_{i=1}^{N} s_i$

C.5.3. Hungarian Centroid Matching for Face Rewards

Algorithm 4 presents our Hungarian centroid matching procedure, which establishes spatial correspondence via centroid distances, then evaluates identity similarity on matched pairs.

C.5.4. Artist Training

Algorithm 5 presents the complete Artist training with GRPO using four compositional rewards and curriculum learning. This is an extension from the explanation in Section 3.4 of the main text.

Algorithm 5 Artist Training (GRPO)

Require: Pre-trained π_{ref} (Flux-Kontext), dataset $\mathcal{T}$, curriculum epoch τ, group size M

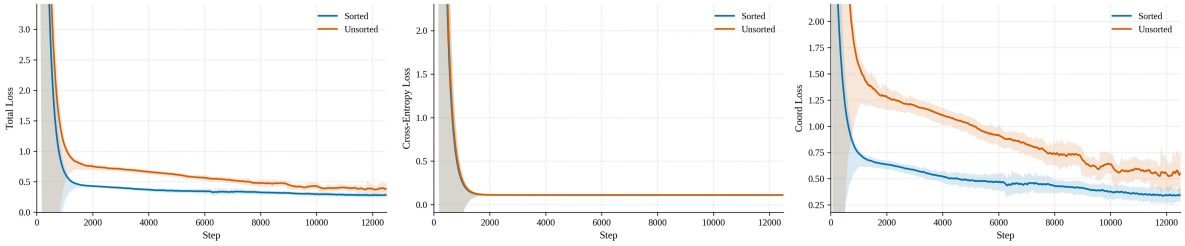
Initialize $\pi_\theta \leftarrow \pi_{\text{ref}}$; partition dataset: $\mathcal{T}_N$ for $N \in \{2, \ldots, 7\}$
for epoch $t = 1$ to N_{epochs} **do**
  Sample from $\mathcal{T}_2 \cup \mathcal{T}_3$ if $t \leq \tau$, else sample uniformly from all $\mathcal{T}_N$
  for each sample $(C, \{I_{\text{ref}}\}, \mathcal{L}, p, I_{\text{target}})$ **do**
    Sample group: $\{x_1, \ldots, x_M\} \sim \pi_\theta(\cdot|p, \mathcal{L}, \{I_{\text{ref}}\})$
    for $i = 1$ to M **do**
      $r_{\text{count}}(x_i) = \mathbb{1}[|\text{RetinaFace}(x_i)| = N]$
      $r_{\text{hps}}(x_i, p) = \text{HPSv3}(x_i, p)$
      $r_{\text{face}}(x_i, \mathcal{L}) = \text{HungarianFaceMatching}(x_i, \mathcal{L}, \{I_{\text{ref}}\})$ (Alg. 4)
      $r_{\text{pose}}(x_i, \mathcal{L}) = \frac{1}{O}\sum_{j=1}^{O} \delta_j$ (frontality scores)
      $r_i = \alpha r_{\text{count}} + \beta r_{\text{hps}} + \zeta r_{\text{face}} + \eta r_{\text{pose}}$
    end for
    Compute advantages: $A_i = (r_i - \mu_G)/(\sigma_G + \epsilon)$
    Update: $\theta \leftarrow \text{Adam}(\nabla \mathcal{L}_{\text{GRPO}})$
  end for
end for
return Fine-tuned Artist

D. Quantitative Results

This section presents additional quantitative analysis and ablation studies that complement our main paper results. We begin by analyzing the impact of coordinate sorting on Architect-A training stability, followed by standalone Architect performance demonstrating the necessity of task-specific training. We then visualize our main results through a radar chart comparison and present our grid search analysis over reward weight configurations. Next, we provide detailed performance breakdowns across varying numbers of people, revealing how methods scale with scene complexity. Finally, we analyze latency and speed comparisons, demonstrating Ar2Can's superior quality-speed trade-off. These analyses provide deeper insights into the effectiveness of our design choices and training strategies.

D.1. Data Sorting for Coordinates Regression

In the LLM-based Architect, we observe that the MLP layers are highly sensitive to the permutation of bounding box coordinate embeddings, which negatively affects performance and convergence stability. To preserve simplicity and scalability in lightweight MLP design, we standardize the input by sorting all coordinates in a consistent order (left to right, top to bottom). This simple yet effective bias improves training stability and enables the model to focus more reliably on coordinate regression. The sorted variant exhibits smoother convergence and overall stronger learning behavior compared to the unsorted setting, as shown in Figure D.1. Compared to the more mature spatial reasoning in the text-to-image framework of Architecture-B, the LLM-based approach in Architecture-A achieves a decent level of spatial structure, indicating that its coordinate representations can be effectively leveraged for the Artist stage, as shown in Figure E.5.



a) Total Loss b) Cross-Entropy Loss c) Co-Ordinate Loss

Figure D.1. **Training curves of Architecture-A (LLM-based) with and without data sorting.** After 2000 steps, training is dominated by the optimization of bounding-box coordinate regression. A clear gap emerges between sorted and unsorted data: sorting leads to more stable learning and faster convergence in the coordinate regression stage under the same number of training iterations.

D.2. Standalone Architect Performance

Table D.1 demonstrates the effectiveness of our training procedures for both Architect variants on count accuracy and spatial diversity. We measure **RMS Spread**, defined as the root mean square of pairwise distances between predicted bounding box centers, which quantifies how well-distributed face locations are across the image (higher values indicate more spatially diverse, natural arrangements). As the prompts differ, a higher spread is ideal. Both Supervised Fine-Tuning (SFT) for Architect-A and Reinforcement Fine-Tuning (RFT) for Architect-B dramatically improve count accuracy over their pretrained baselines. Architect-A achieves 97.7% (from 15.2%) and Architect-B reaches 93.2% (from 59.9%). Notably, Architect-B produces higher RMS Spread (0.07) than Architect-A (0.05), reflecting its 2D T2I backbone's stronger spatial priors for distributing people naturally across the scene.

D.3. Grid Search on Reward Weights

Table D.2 presents our grid search over reward weight configurations. Due to compute constraints, we evaluate at 150 training epochs, for a 7-GPU run. Each reward component offers distinct benefits: **Face matching** (ζ) provides the strongest identity preservation signal (84.9 Multi-ID when isolated), but sacrifices count accuracy and prompt alignment. The resulting images look poor. We show them in Figure E.7. Hence, we need a high quality reward signal such as **HPSv3** (β). It is critical for both realism and prompt alignment, with higher weights improving aesthetic quality but potentially overwhelming identity signals. This is also visible in Figure E.7. Next, we increase **Count** reward weight (α), which directly improves counting but can conflict with natural scene composition. Finally, **Frontal Pose** (η) reduces copy-paste artifacts by encouraging natural face orientations.

Model	Count Accuracy↑	RMS Spread↑
MultiHuman-TestBench		
Qwen-2.5-0.5B (Baseline)	15.2	0.01
Qwen-2.5-0.5B + SFT (Architect-A)	97.7	0.05
Flux-Schnell (Baseline)	59.9	0.04
Flux-Schnell + RFT (Architect-B)	93.2	0.07

Table D.1. Count accuracy and spatial diversity (RMS Spread) of Architect variants on MultiHuman-Testbench. RMS Spread measures the root mean square of pairwise distances between predicted bounding box centers; higher values indicate more spatially diverse layouts. Both training procedures significantly improve count accuracy over their baselines, with Architect-B producing more spread-out spatial arrangements owing to its 2D generative backbone.

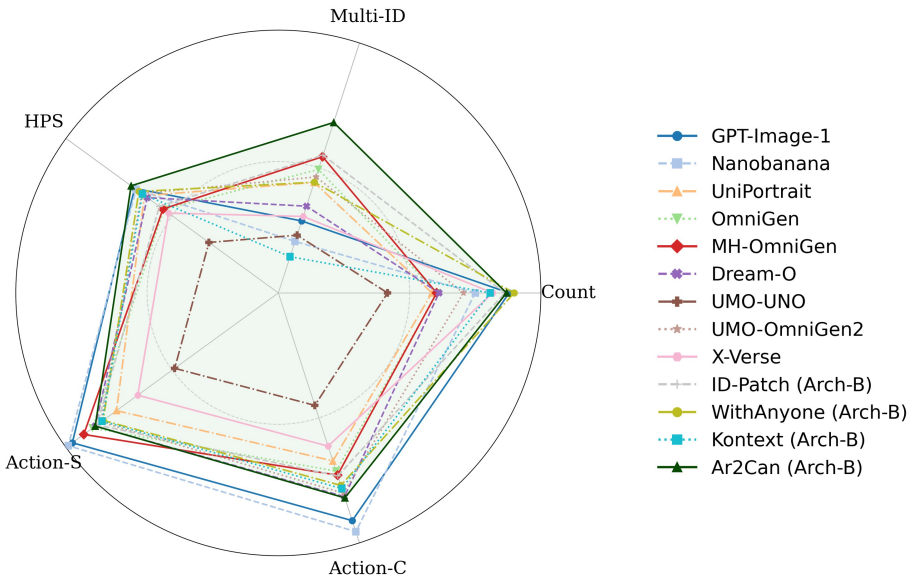


Figure D.2. Radar chart visualization of results from Table 1. Ar2Can achieves the most balanced performance across all MultiHuman-Testbench metrics, demonstrating superior count accuracy and identity preservation while maintaining competitive prompt alignment and action scores. The SOTA methods exhibit clear trade-offs, excelling in some metrics while failing in others.

As observed in Table D.2, our selected configuration in blue ($\alpha = 0.2, \beta = 0.4, \zeta = 0.3, \eta = 0.1$) achieves the best balance: strong count accuracy (84.5), high identity preservation (65.9), and competitive prompt alignment (29.2). This weighting prioritizes HPSv3 for realism while maintaining sufficient face matching signal for identity preservation, with modest count and pose rewards to prevent failure modes. Extreme configurations (e.g., $\zeta = 1.0$) achieve higher single-metric performance but fail to generalize across evaluation dimensions.

Model	Reward Weights				Metrics		
	Count α	HPSv3 β	Face ζ	Pose η	Count Acc.	Multi-ID	Prompt Align
Baseline	0	0	0	0	80.7	14.5	29.2
	0	0	1	0	72.7	84.9	25.6
	0	0.5	0.5	0	82.3	72.2	28.6
	0.2	0.6	0.2	0	85.3	35.2	29.9
	0.2	0.3	0.3	0.2	84.3	58.2	28.4
Ar2Can	0.2	0.4	0.3	0.1	84.5	65.9	29.2

Table D.2. Grid search on Artist reward weights at 150 epochs. Each reward offers distinct benefits: face matching (ζ) maximizes identity preservation, HPSv3 (β) enhances realism and prompt alignment, count (α) improves accuracy, and pose (η) reduces copy-paste artifacts. Our selected configuration balances all objectives. Performance differs from main paper due to limited training (fewer epochs/GPUs).

D.4. Performance across varying number of people

Table D.3 presents a detailed breakdown of Multi-ID similarity and person count accuracy across different group sizes (2-5 people) on MultiHuman-Testbench. This analysis reveals how methods scale with increasing scene complexity. Most baseline methods exhibit severe performance degradation as the number of people increases, particularly in identity preservation. For instance, MH-OmniGen's Multi-ID drops from 65.3 (2 people) to 38.0 (5 people), while count accuracy falls from 91.2 to 19.7. UniPortrait and OmniGen show similar collapse patterns for larger groups. In contrast, Ar2Can maintains consistently high performance across all person counts, with Multi-ID scores remaining stable (67.7 to 69.5) and count accuracy actually improving with more people (85.1 to 90.9). This demonstrates that our two-stage architecture with explicit spatial grounding effectively prevents the identity merging and counting failures that plague end-to-end methods in complex multi-human scenarios.

Model	Multi-ID Similarity					Person Count Accuracy				
	2	3	4	5	Avg	2	3	4	5	Avg
MultiHuman-Testbench										
GPT-Image-1	31.8	29.5	27.8	24.9	28.8	90.7	91.8	89.5	75.3	87.9
Nanobanana	21.8	17.5	11.8	10.4	20.6	84.0	81.1	71.5	55.7	75.0
Fastcomposer	15.3	7.4	7.2	5.9	12.2	62.9	11.2	3.2	1.1	31.2
UniPortrait	56.5	46.4	33.8	28.6	44.2	90.6	76.3	23.7	14.1	58.5
OmniGen	60.8	52.3	42.2	35.2	49.4	88.8	88.0	23.2	21.6	60.5
MH-OmniGen	65.3	60.4	45.1	38.0	54.5	91.2	87.5	22.4	19.7	60.3
Dream-O	48.7	30.9	20.0	15.9	34.7	92.3	86.1	26.9	15.2	61.2
UMO-OmniGen2	56.6	49.2	36.8	30.6	46.4	92.0	93.6	41.3	40.0	70.5
X-Verse	39.6	33.9	24.6	20.7	30.6	96.9	95.1	90.1	40.0	81.7
WithAnyone	45.7	44.4	44.0	42.7	44.3	90.4	91.5	89.6	88.3	89.8
Ar2Can (Ours)	67.7	71.8	71.7	69.5	67.6	88.1	86.9	89.9	90.9	90.2

Table D.3. Performance breakdown across different group sizes (2-5 people) on MultiHuman-Testbench. Color coding: highest and lowest. Ar2Can (Architect-A) maintains consistently high Multi-ID similarity and count accuracy across all person counts, while baseline methods exhibit severe degradation with increasing scene complexity.

D.5. Latency Analysis and Speed Comparisons

Table D.4 provides a detailed breakdown of inference times for Ar2Can and state-of-the-art methods on A100 GPU at 1024×1024 resolution with 3 identities. We separately report the Architect inference time (layout generation), Artist inference time (image rendering), and total latency for our two-stage approach.

As observed, Ar2Can achieves the best quality-speed trade-off among all methods. Architect-A (Qwen-based) requires only 0.5 seconds for layout generation, while Architect-B (Flux-Schnell based) takes 1.4 seconds. The Artist with token sharing reduces inference time by approximately $2\times$ compared to the full canvas baseline (15s vs 28s), while maintaining the highest unified scores (72.4 and 72.2). Architect-A offers the fastest overall latency (15.5s) with the highest unified score (72.4), while Architect-B provides slightly lower but still competitive performance (72.2) with a total latency of 16.4s.

Compared to other methods, Ar2Can with token sharing (Arch-A) achieves the second fastest total latency (15.5s) among all methods while delivering significantly higher quality (72.4 unified score). Methods like UMO-OmniGen2 and X-Verse require substantially longer inference times (74s and 87s respectively) while achieving lower unified scores. This demonstrates that our modular architecture with efficient token sharing provides superior performance without sacrificing speed.

D.6. Unified Architecture Variant (Architect-C)

To validate our choice of two lightweight specialist Architects, we introduce **Architect-C**, which finetunes BAGEL [9], which is a unified multimodal large language model. For this finetuning, we use the same GRPO procedure as Architect-B (Section 3.3.2). Table D.5 compares all three variants across standalone layout quality and final Artist performance. Architect-C achieves competitive results (ID: 68.0, HPS: 30.9, RMS Spread: 0.09) comparable to Architect-A and Architect-B, confirming that a unified model can capture both language understanding and spatial reasoning under our RL training. It obtains the best RMS spread and HPS which signifies that this has the best spatial layout compared to other methods. However, it incurs a 27s layout generation time. This is roughly $19\times$ slower than Architect-A (0.5s) and Architect-B (1.4s),

Model	Diffusion Steps	Architect (sec)	Artist (sec)	Total (sec) ↓	Unified Score ↑
MH-OmniGen	50	—	59	59	61.6
UMO-OmniGen2	50	—	74	74	60.4
Dream-O	25	—	57	57	59.7
X-Verse	50	—	87	87	52.7
WithAnyone (Arch-B)	25	1.4	14	15.4	62.6
Ar2Can-Full (Arch-B)	28	1.4	28	29.4	71.5
Ar2Can-Shared (Arch-A)	28	0.5	15	15.5	72.4
Ar2Can-Shared (Arch-B)	28	1.4	15	16.4	72.2

Table D.4. Latency breakdown on A100 GPU (1024×1024, 3 identities). Color coding: best and lowest. Arrows indicate optimization direction: ↓ lower is better, ↑ higher is better. Ar2Can's Architect-A achieves the fastest total latency (15.5s) with the highest unified score (72.4), demonstrating superior quality-speed trade-off. Token sharing accelerates the Artist by approximately $2\times$ (15s vs 28s full canvas).

making it impractical for deployment despite marginally higher layout diversity. This confirms that our two lightweight specialists provide a superior efficiency v/s quality tradeoff.

Architect	Architect Performance			Our Artist (Final)		
	Acc.↑	RMS Spread↑	Time↓	Acc.↑	ID↑	HPS↑
Architect-A	97.7	0.05	0.5s	90.2	67.6	30.2
Architect-B (Schnell)	93.2	0.07	1.4s	86.9	68.2	30.8
Architect-C (BAGEL)	96.1	0.09	27s	88.2	68.0	30.9

Table D.5. Comparison of Architect variants on standalone layout quality and final Artist performance. **Spread**: RMS spread of bounding box centers (higher = more spatially diverse layouts). Color coding: best and worst per column.

D.7. Off-the-Shelf Layout Generation as Architect

As an alternative to task-specific Architect training, one could use off-the-shelf layout generation methods such as LayoutGPT [14] and RPG-DiffusionMaster [50] as drop-in Architects, feeding their predicted layouts directly to our Artist. We use the method proposed by these works to generate boxes, using both GPT-4o and Llama prompted to generate boxes. The results in Table D.6 show that while RPG/LayoutGPT with GPT-4o achieves high count accuracy (100.0), both methods produce very similar layouts for every prompt (RMS-Spread: 0.01-0.02). This directly degrades final Artist performance (ID: 57.2 for GPT-4o vs. 67.6 for Architect-A), demonstrating that layout *diversity* is as critical as count accuracy, and that task-specific Architect training is necessary.

Architect	Backbone	Architect Performance			Our Artist (Final)		
		Acc.↑	Spread↑	Time↓	Acc.↑	ID↑	HPS↑
RPG/LayoutGPT	GPT-4o	100.0	0.01	14s	78.6	57.2	27.5
RPG/LayoutGPT	Llama	51.0	0.02	3s	45.3	31.3	27.0
Architect-A	Qwen-2.5-0.5B	97.7	0.05	0.5s	90.2	67.6	30.2
Architect-B	Flux-Schnell	93.2	0.07	1.4s	86.9	68.2	30.8

Table D.6. Off-the-shelf layout generators vs. our trained Architects as drop-in replacements feeding the same Artist. Despite high count accuracy, RPG/LayoutGPT produces low-diversity layouts (Spread) that substantially degrade final generation quality. Color coding: best and worst per column.

E. Qualitative Results

This section presents comprehensive qualitative analysis and visual comparisons that illustrate the effectiveness of our proposed components. We begin with extensive qualitative comparisons on the Multi-ID Test benchmark, demonstrating Ar2Can's ability to jointly optimize identity preservation and prompt alignment. We then showcase results from our optional pose-controlled Artist variant and visualize the effectiveness of our token sharing strategy for handling overlapping regions. Additional comparisons demonstrate the advantages of Hungarian centroid matching over naive spatial matching, the complementary strengths of our two Architect variants, the validity of our frontal pose scoring, and the critical role of HPSv3 in enhancing aesthetic quality.

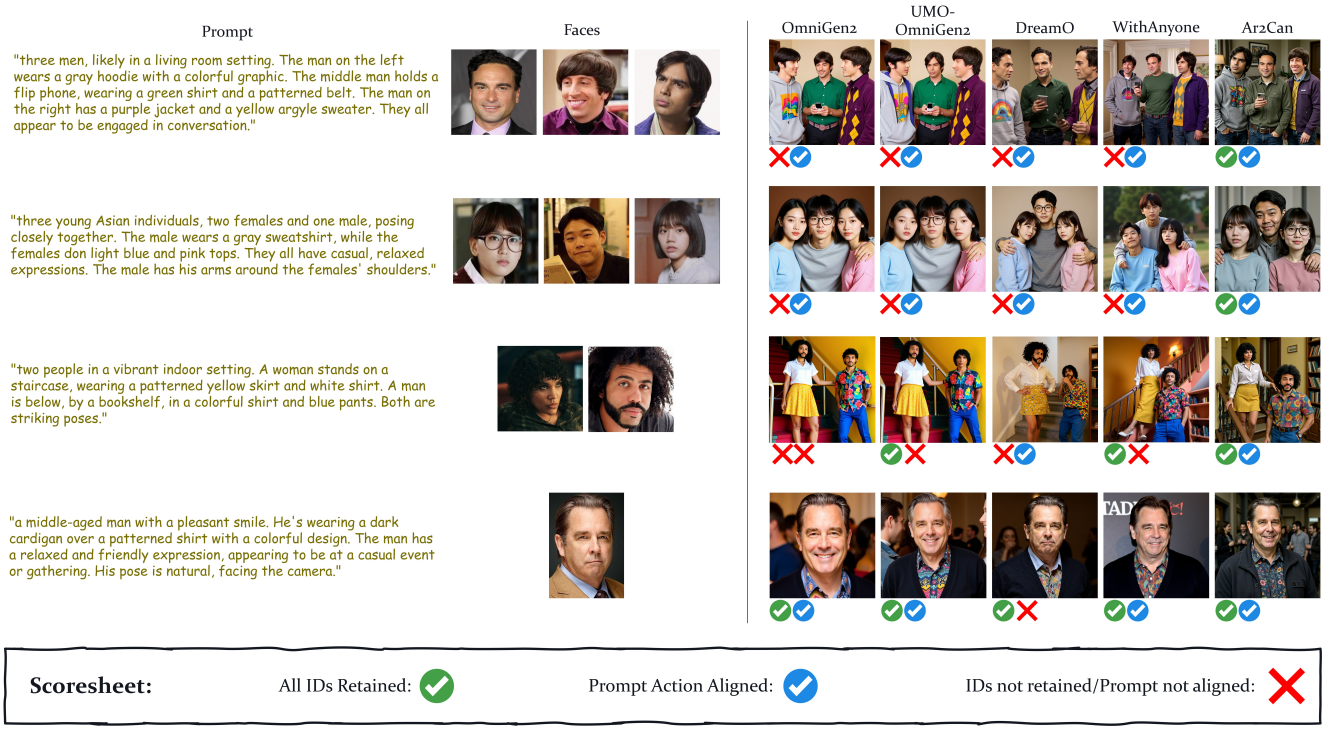


Figure E.1. Qualitative comparison with state-of-the-art methods on Multi-ID Test. Each row shows reference identities and generations from multiple methods given highly descriptive prompts specifying clothing, expressions, item placement, and spatial arrangements. Ar2Can consistently preserves all input identities while accurately following detailed prompt specifications. Baseline methods exhibit a trade-off: either achieving identity preservation at the cost of prompt alignment, or vice versa. This limitation becomes more severe as the number of people increases.

E.1. Qualitative Comparison on Multi-ID Test

Figure E.1 presents a comprehensive qualitative comparison between Ar2Can and state-of-the-art methods on the Multi-ID Test benchmark. Each row shows a different multi-person scenario with highly descriptive prompts that specify fine-grained details including clothing styles, facial expressions, item placement, and spatial arrangements.

As observed across all examples, Ar2Can consistently achieves both objectives simultaneously: (1) faithful identity preservation for every input face, and (2) accurate alignment with detailed prompt specifications. In contrast, existing methods exhibit a clear trade-off between these objectives. Methods optimized for identity preservation often fail to execute prompt-specified details correctly. They miss clothing attributes, incorrect expressions, or wrong item placements. Conversely in cases where methods achieve better prompt alignment, frequently suffer from identity hallucination or blending.

This failure to jointly optimize both objectives becomes more pronounced as the number of people increases. For scenes with 3 people, baseline methods systematically fail at either identity consistency or prompt adherence. Ar2Can has a two-stage architecture, and the artist's GRPO-based training method contains explicit spatial grounding and Hungarian-based face matching. This enables it to maintain both high identity fidelity and precise prompt alignment across varying person counts.

E.2. Pose-Controlled Artist Results

Figure E.2 demonstrates the capabilities of our pose-controlled Artist variant, trained with the pose reward described in Section C. This variant conditions on both face bounding boxes and human pose skeletons overlaid on the canvas, enabling explicit control over body poses and actions in the generated images.

As observed in the results, the pose-controlled Artist successfully achieves multiple objectives: (1) faithful identity preservation for all input faces across different scenes, (2) accurate alignment with the specified input poses, and (3) photorealistic rendering quality with natural lighting and coherent scene composition. The model demonstrates the ability to generate diverse multi-person scenarios while maintaining precise pose control. It can generate images ranging from complex group poses to dynamic action sequences, all while preserving the distinctive facial features of each reference identity.

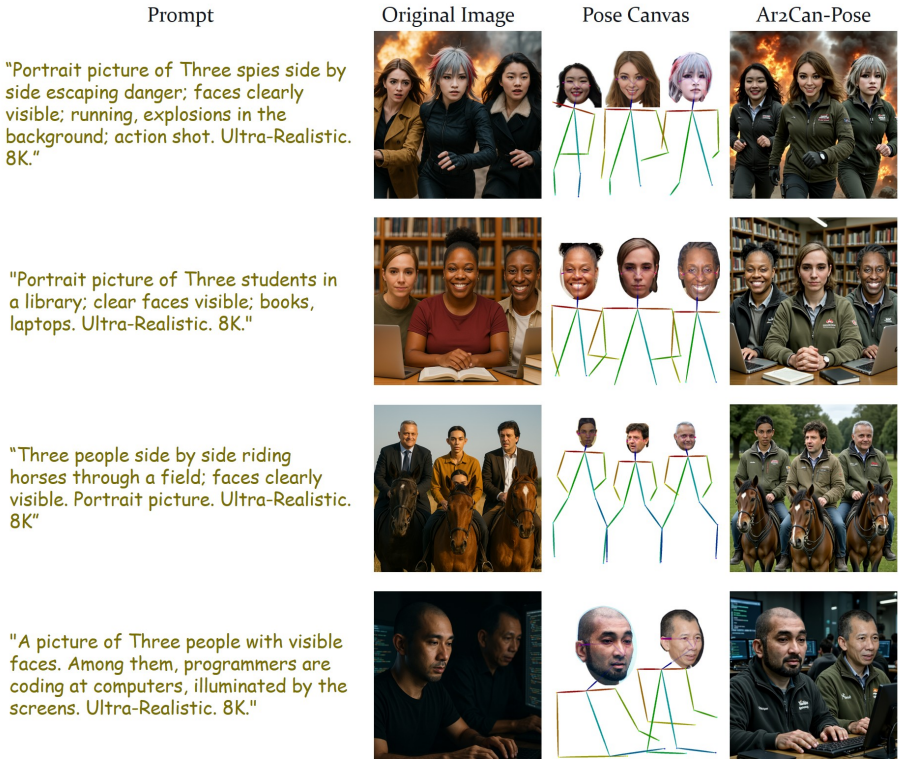


Figure E.2. Results from our pose-controlled Artist variant trained with pose rewards. The model successfully preserves all input identities while accurately aligning with specified input poses, generating photorealistic multi-person scenes with controlled actions and body configurations.

However, we observe an increase in copy-paste artifacts in this variant compared to our standard Artist. The additional constraint of matching specific body poses appears to occasionally lead to more rigid spatial composition, where faces and bodies are sometimes rendered with less natural integration into the scene context. This represents a trade-off between pose controllability and rendering flexibility. We note that these results represent an initial exploration of pose-controlled multi-human generation. The pose-controlled Artist demonstrates the feasibility of fine-grained action control while maintaining identity preservation, but addressing the copy-paste artifacts requires further investigation. We are actively working on improving this variant. This direction represents promising future work for extending Ar2Can's capabilities to fine-grained controllable generation.

E.3. Effectiveness of Token Sharing

In Figure E.3, we demonstrate the effectiveness of our proposed token sharing strategy for handling overlapping spatial regions. The figure shows four columns: the prompt, canvas input, generation without shared tokens, and generation with shared tokens.

As observed in the results, the shared token approach enables the model to naturally determine depth ordering between people. When multiple face regions overlap, the model learns to resolve spatial conflicts through intelligent compositional strategies. It either arranges people with appropriate depth layering or by spatially reorganizing them to avoid unnatural overlap. Hence, the results with shared tokens are perceptually better, exhibiting more natural poses, realistic occlusions, and coherent spatial arrangements. Importantly, this approach is both automatic and controllable. When no explicit ordering is desired, the shared positional encodings allow the model to determine the most natural composition. However, **when specific depth control is needed** (e.g., to place a particular person in the foreground), we can selectively disable shared encodings for the region corresponding to the person who should appear behind. This provides users with fine-grained control over depth ordering while maintaining the perceptual benefits of natural scene composition.

Figure E.3. Visual comparison demonstrating the effectiveness of token sharing. **Left**: Canvas input with overlapping face regions. **Middle**: Generation without shared tokens, resulting in unnatural spatial conflicts. **Right**: Generation with shared tokens, enabling natural depth ordering and realistic occlusions. The shared token approach produces perceptually superior results while maintaining controllability when explicit depth ordering is required.

E.4. Naive Face Matching vs. Hungarian Centroid Face Matching

Figure E.4 provides visual evidence for the effectiveness of our Hungarian Centroid Matching (HCM) approach compared to naive location-based matching. As shown in Table 3 of the main paper, simple matching improves Multi-ID scores (55.2) but significantly degrades count accuracy (80.7→75.6) and image quality (HPS: 29.2→27.6).

The visual results clearly illustrate the underlying problem: simple matching, which extracts faces at exact predicted bounding box locations, produces copy-paste artifacts and faces with unnatural sizes. The rigid constraint of exact spatial localization forces the model to paste reference faces directly at specified coordinates, often resulting in faces that are too large, too small, or incorrectly scaled relative to the body and scene context. These artifacts create perceptually jarring images that lack photorealism despite achieving reasonable identity similarity.

In contrast, Hungarian Centroid Matching relaxes these rigid spatial constraints by matching faces based on centroid proximity rather than exact bounding box overlap. This flexibility allows the model to adjust face sizes and precise locations based on prompt requirements and aesthetic considerations. The model can now render faces with natural scaling, appropriate depth cues, and realistic proportions relative to bodies and the overall scene composition. As demonstrated in the ablation study (Table 3), HCM recovers image quality (HPS: 30.9) while further improving Multi-ID scores (60.3). Hence, it achieves the best balance between spatial accuracy, identity preservation, and photorealism.

E.5. Architect Variant Comparison

Figure E.5 presents a visual comparison of canvases generated by our two Architect variants given identical text prompts. The figure illustrates the fundamental differences in spatial layout generation between Architect-A (LLM-based, left) and Architect-B (T2I-based, right).

As observed in the results, Architect-B produces more spread out face arrangements with greater spatial diversity in the distribution of people across the canvas. The 2D nature of the T2I backbone enables it to naturally capture complex spatial relationships and varied compositional layouts, often placing people at different depths and with more dynamic spatial configurations. This results in naturally varied scene compositions.

In contrast, Architect-A generates layouts with more compact and structured face arrangements. While these layouts may appear less spatially diverse, Architect-A demonstrates superior count accuracy due to its strong language understanding capabilities, which enable it to reliably parse numerical references from text prompts. Additionally, Architect-A achieves significantly faster inference times (1.4s vs 0.5s as shown in Table D.4), making it more suitable for latency-sensitive applications.

Figure E.4. Visual comparison between naive location-based matching and Hungarian Centroid Matching (HCM). **Simple Matching** enforces exact spatial localization, leading to copy-paste artifacts and unnatural face sizes that degrade perceptual quality. **Hungarian Centroid Matching** relaxes spatial constraints by matching on centroid proximity, enabling the model to flexibly adjust face sizes and positions based on prompt semantics and aesthetic considerations. This produces photorealistic results with natural proportions while maintaining strong identity preservation.

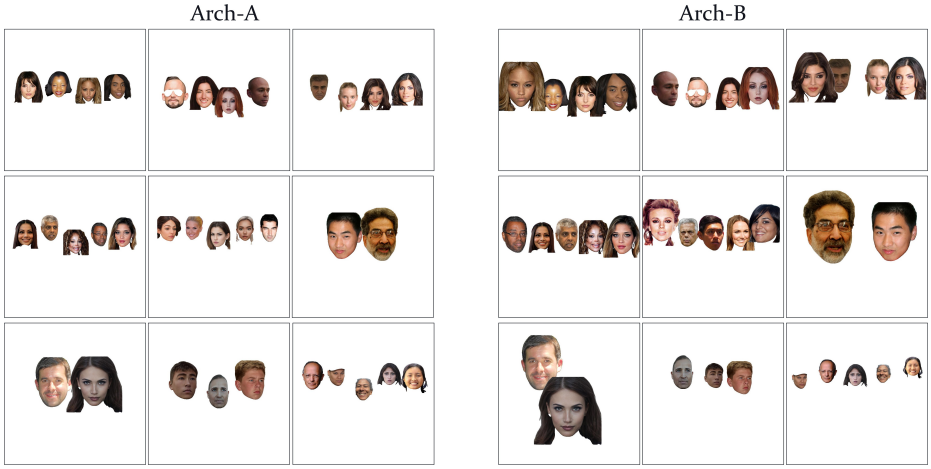


Figure E.5. Visual comparison of canvases generated by Architect-A (left) and Architect-B (right) given identical prompts. Architect-B produces more spatially diverse layouts with spread out face arrangements and varied distributions, leveraging its 2D T2I backbone to capture complex spatial relationships. Architect-A generates more compact, structured layouts with superior count accuracy and faster inference times. Each variant offers complementary strengths for different deployment scenarios.

Each Architect variant offers distinct advantages: Architect-B excels at generating spatially rich and varied layouts with natural pose information, while Architect-A provides more reliable count accuracy with faster generation speeds. This

modular design allows users to select the appropriate Architect based on their specific requirements.

E.6. Visualizing Frontal Pose Scores

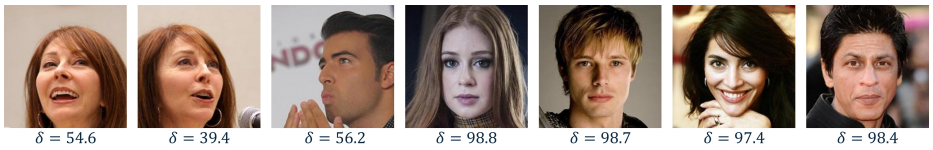


Figure E.6. Frontal pose score validation. The keypoint-based scoring method assigns high scores (> 0.9) to forward-facing faces and low scores to rotated or tilted faces, proving as an effective reward for encouraging natural camera-facing poses during training.

Figure E.6 illustrates our frontal pose scoring on sample faces from the test set. The frontal score effectively distinguishes frontal faces (score > 0.9) from non-frontal faces (significantly lower scores). Hence, we use it as a reward signal to reduce copy-paste artifacts.

E.7. HPSv3 Reward Impact

Impact on Architect-B. In Architect-B, removing HPSv3 from the GRPO reward causes the model to reward-hack the count objective by predicting repetitive, clustered layouts regardless of the input prompt. Similar to the analysis shown in Table D.7, RMS Spread drops from 0.07 to 0.02, producing very similar spatial arrangements in every image. HPSv3 prevents this by enforcing prompt-aware, spatially diverse layouts.

Figure E.7. Impact of HPSv3 reward on generation quality. Without HPSv3, images exhibit flat lighting, unnatural colors, and poor compositional balance despite preserving identities.

Impact on Artist. Figure E.7 demonstrates the dual benefit of the HPSv3 reward in Artist training. Beyond improving prompt alignment, HPSv3 significantly enhances overall aesthetic quality, producing more photorealistic lighting, natural skin tones, coherent scene composition, and professional-grade rendering. Visual comparison shows that without HPSv3, generated images suffer from flat lighting, unnatural colors, and poor compositional balance, even when identity preservation is maintained. This validates HPSv3 as a critical component for achieving both semantic accuracy and visual realism in multi-human generation.

E.8. Generalization to Multi-Object and Multi-Human+Object Scenes

Figure E.8. Occlusion handling in multi-object scenes. Given an input prompt, canvas, and output image, our token sharing scheme with shared RoPE encodings naturally resolves spatial conflicts between overlapping objects, producing realistic depth ordering and occlusions without explicit supervision.

A notable property of our Artist is its ability to generalize beyond multi-human generation to multi-object and mixed multi-human+object scenes, **without any retraining or fine-tuning on object data**. As shown in Figure E.9, given a canvas with multiple object and human reference images pasted, the Artist synthesizes photorealistic outputs with natural scene composition. This is a strong result, as the model was never exposed to object-centric training data. The spatially-grounded rewards and compositional training generalize naturally to arbitrary identity-preserving generation beyond faces. Beyond simple placement, the Artist demonstrates several emergent compositional capabilities: (1) **appearance editing**, such as adding sunglasses to plush toys, with the edit naturally blended into the scene lighting and environment; (2) **object interaction**, where multiple objects relate to one another coherently within the scene; and (3) **depth ordering**, where as shown in Figure E.8, our token sharing scheme with shared RoPE encodings naturally resolves spatial conflicts between overlapping objects, producing realistic occlusions and layering. These results suggest that our model learns generalizable compositional capabilities that extend well beyond the multi-human domain.

Figure E.9. Multi-human+object generation. Given an input prompt and reference images for both people and objects, we construct a canvas by pasting segmented references at Architect-predicted locations. The Artist synthesizes photorealistic outputs preserving all input identities while naturally composing humans and objects within the scene.

F. Limitations and Future Work

While Ar2Can achieves state-of-the-art performance on multi-human generation, some limitations remain for future exploration. As the number of identities scales beyond 3 people, we observe reduced control over individual facial expressions and body poses. Although our method supports prompt-based editing for fine-grained attributes such as clothing, hairstyles, and accessories across all group sizes, per-person expression control in larger groups ($>$ 3 people) remains challenging. Additionally, the current approach struggles with complex multi-person poses and interactions, as reflected in our Action-C scores, which show room for improvement compared to simpler action scenarios. Future work will focus on: (1) extending fine-grained expression and pose control to larger groups, (2) improving complex action generation through enhanced pose conditioning or hierarchical scene composition, and (3) exploring scalability to even larger group sizes (8+ people) while maintaining individual control. These directions will enable more versatile and controllable multi-human generation for diverse applications.

G. Disclosure of LLM Use

We used large language models (LLMs) for editing assistance, including grammar correction, language refinement, and vocabulary improvement. We also used LLMs for generating training prompts as described in Appendix B. All scientific ideas, technical contributions, experimental designs, and conclusions presented in this paper are entirely our own.