INTER IIT Tech Meet 2025

Team ID-24

ABSTRACT This project captures our journey toward building an automated version of Photoshop that can understand user intent and perform edits on its own. Our focus was to create a system that is accurate, fast, and lightweight enough to run on everyday edge devices. Along the way, we experimented with different models, refined our pipelines, and solved several challenges related to efficiency, visual quality, and deployment. This report highlights the ideas we explored, the technologies we used, and the improvements we made as we moved closer to a practical, on-device photo-editing assistant.

I. INTRODUCTION

IN this report, we present our work on automating core Photoshop operations using efficient vision and generative models. Our aim was to build something that feels intuitive for users while being powerful enough to handle tasks like object removal, relighting, and replacement with minimal effort. Throughout the project, we kept two things in mind: maintaining strong editing quality and ensuring the entire system could comfortably run on edge devices. We tested multiple architectures, experimented with different approaches, and made several optimizations to balance accuracy with performance. What follows is a clear summary of our experiments, the challenges we faced, and the solutions that shaped the final system.

II. RELATED WORK

Our segmentation component builds on the Segment Anything Model family, including SAM-Large (ViT-H), SAM-Huge (ViT-L), and SAM-Base (ViT-B), which provide strong promptable masks but are slow on CPU. For mobile deployment, we rely on MobileSAM (Tiny-ViT SAM), which preserves SAM's decoder while significantly reducing compute. For text-guided mask selection, we use MobileCLIP, a lightweight CLIP variant that enables efficient text–image similarity scoring and supports our spatial-plus-semantic mask ranking strategy, similar to proposal filtering found in Mask2Former, Mask R-CNN, DETIC, ReferFormer, and LLaVA-Grounding.

For object removal, we use the dilated LaMa inpainting model, which employs dilated and frequency-domain convolutions to achieve high-quality hole filling. Earlier approaches such as partial convolutions, EdgeConnect, and RFR-Net form the foundation of this class of techniques. We further apply FP16 and INT8 post-training quantization to improve LaMa's latency while maintaining reconstruction quality on CPU.

Our relighting pipeline relies on MiDaS-small for monocular depth estimation, providing the depth maps necessary for surface-normal computation. The relighting formulation follows classical Lambertian shading with a rim-light prior, and a FiLM-conditioned U-Net refines the illumination. Training supervision is provided by the VIDIT-Depth-ControlNet dataset, which contains paired images under varying lighting conditions.

For object and background replacement, we use Stable Diffusion 1.5 Inpainting. To enable CPU-friendly sampling, we integrate LCM adapters, reducing generation to 5–8 steps, and LoRA adapters to enhance texture and lighting consistency under low-step diffusion. Finally, a lightweight conversational layer built on Gemini Flash 2.5 maps natural-language instructions into structured commands—task type, segmentation prompt, and diffusion prompt—enabling automated routing across all modules.

III. AUTOMATED EDITING PIPELINES

This section summarizes all ML components used in our system. We group the core vision–language and generative modules into Tasks 1–5.

A. TASK 1: SEGMENTATION

The first task in our system involves generating high-quality object masks that enable object removal, matte extraction, and region-specific editing on CPU-only edge hardware. The primary challenge is achieving accurate segmentation while maintaining latency low enough for interactive use. To this end, we evaluated multiple variants of the Segment Anything Model (SAM) to identify a backbone that offers the optimal trade-off between segmentation fidelity, computational cost, and compatibility with prompt-based editing workflows.

1) Methodology

We benchmarked three official SAM variants—SAM-Large (ViT-H), SAM-Huge (ViT-L), and SAM-Base (ViT-B)—followed by the MobileSAM family designed for lightweight deployment. Each model was evaluated using point- and box-prompt inputs under identical CPU-only conditions. Performance metrics included segmentation quality (mask precision, boundary consistency) and end-to-end latency measured per image.

SAM-Large (ViT-H, ∼630M parameters) delivered the strongest mask quality but incurred an average latency of ∼45 s/image, making it impractical. SAM-Base (ViT-B, ∼90M parameters) reduced latency to ∼23 s/image, while SAM-Huge (ViT-L, ∼300M parameters) achieved the best CPU performance among the original models at ∼9 s/image; however, all variants remained unsuitable for real-time operation.

MobileSAM replaces the standard ViT backbone with a Tiny-ViT encoder (5M parameters) while retaining SAM's mask decoder. This preserves compatibility with SAM's prompt interface but drastically reduces computational cost. Under identical conditions, MobileSAM achieved an average inference latency of ∼1.45 s/image, with only minor quality degradation on fine object boundaries.

2) Results

MobileSAM was the only model satisfying all deployment constraints: (1) mask quality sufficient for downstream diffusion-based inpainting, (2) CPU latency compatible with interactive editing, (3) small model footprint for edge devices, and (4) full compatibility with SAM's prompt-driven segmentation paradigm. Consequently, MobileSAM was selected as the primary segmentation backbone for our object-level editing pipeline.

B. TASK 2: PROMPT-BASED SEGMENTATION

This task performs segmentation directly from natural-language prompts while operating under mobile and CPU-only constraints. Instead of relying on heavy vision–language segmentation models, we introduce a spatially aware, text-driven pipeline combining lightweight mask proposals with semantic reasoning. The system outputs a single binary mask m^* aligned with the user's linguistic intent and spatial reference in the scene.

1) Methodology

System Overview. Given image I and prompt P, the system operates in three stages: (1) mask proposal generation, (2) spatial intent filtering, and (3) semantic ranking. The final mask maximizes spatial and semantic alignment.

Mask Proposal Generation. MobileSAM produces a set of class-agnostic proposals

$$\mathcal{M} = \{m_1, \ldots, m_N\}.$$

For each mask m_i, we compute its centroid using image moments:

$$(x_i, y_i) = \left(\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}}\right),$$

and normalize it as

$$\hat{C}_i = \left(\frac{x_i}{W}, \frac{y_i}{H}\right).$$

We also compute a normalized area ratio:

$$R^{(i)}_{\text{area}} = \frac{1}{HW}\sum m_i.$$

Spatial Intent Extraction & Filtering. Directional cues (e.g., "left", "right", "top", "bottom") in P map to a coarse target coordinate μ_{target}. Spatial relevance is measured by:

$$d_i = \|\hat{C}_i - \mu_{\text{target}}\|_2, \qquad W^{(i)}_{\text{spatial}} = \exp\left(-\frac{d_i^2}{2\sigma^2}\right).$$

Masks with weight below a threshold are discarded:

$$\mathcal{M}' = \{m_i \in \mathcal{M} \mid W^{(i)}_{\text{spatial}} \geq \tau\}.$$

Semantic Scoring via MobileCLIP. For each $m_i \in \mathcal{M}'$, we extract the masked crop $I^{(i)}_{\text{crop}}$ and obtain normalized embeddings:

$$\mathbf{v}_i = \frac{E_{\text{img}}(I^{(i)}_{\text{crop}})}{\|E_{\text{img}}(I^{(i)}_{\text{crop}})\|}, \quad \mathbf{t} = \frac{E_{\text{txt}}(P)}{\|E_{\text{txt}}(P)\|}.$$

Semantic similarity is the cosine score:

$$S^{(i)}_{\text{sem}} = \mathbf{v}_i \cdot \mathbf{t}.$$

Context-Aware Optimization. To handle ambiguous or overlapping proposals, we combine semantic, spatial, and area cues:

$$S^{(i)}_{\text{final}} = \alpha S^{(i)}_{\text{sem}} + \beta W^{(i)}_{\text{spatial}} + \gamma R^{(i)}_{\text{area}}.$$

The selected mask is:

$$m^* = \arg\max_{m_i \in \mathcal{M}'} S^{(i)}_{\text{final}}.$$

2) Results

The spatially aware segmentation pipeline substantially reduces computational load by filtering out spatially inconsistent masks before semantic scoring, lowering MobileCLIP calls by up to an order of magnitude. The combination of spatial cues, semantic similarity, and geometric priors yields masks that consistently match user intent, even in cluttered scenes, enabling efficient and accurate prompt-based object selection on CPU-only edge devices.

FIGURE 1. Results for Text based Segmentation with an approximate latency of 14 seconds.

C. TASK 3: OBJECT REMOVAL

Once the target region is segmented and removed, the system must reconstruct the missing content in a manner that is visually coherent, texture-consistent, and computationally efficient for mobile deployment. After evaluating multiple inpainting architectures, the LaMa family emerged as the only viable solution offering both high perceptual quality and acceptable CPU performance. Our final design adopts the dilated LaMa model as the core inpainting backbone, supplemented with model quantization to further optimize latency.

1) Methodology

The dilated LaMa variant leverages fast Fourier convolutions and large receptive fields obtained through dilated kernels, enabling it to synthesize globally consistent structure while preserving fine textures. Among the four LaMa configurations we tested, the dilated model delivered the best trade-off between accuracy and efficiency, achieving an average latency of approximately $0.54\,\mathrm{s}$ per image on CPU while maintaining a compact model size suitable for memory-constrained devices. In contrast, the Fourier-based LaMa variant produced sharper and more globally coherent reconstructions but required $\sim 3.4\,\mathrm{s}$ per image, rendering it unsuitable for real-time, on-device execution.

To further accelerate inference, we applied post-training quantization to the dilated LaMa model. FP16 quantization reduced compute cost with no observable degradation in visual fidelity and improved latency by 10–15%, becoming the default configuration for CPU deployment. INT8 quantization achieved even lower computational load, introducing only minimal quality loss due to LaMa's robust architectural priors, and remains an option for extremely resource-limited hardware.

2) Results

The final inpainting workflow proceeds as follows: (1) receive the selected mask from the segmentation module, (2) apply the quantized dilated LaMa model to reconstruct the masked region, and (3) seamlessly blend the generated content with the original image using lightweight edge smoothing. This pipeline consistently produces sharp boundaries, texture-coherent fills, and stable performance across diverse scenes, achieving an average CPU latency of $\sim 0.5\,\mathrm{s}$. These characteristics make the system well suited for real-time, fully on-device editing applications.

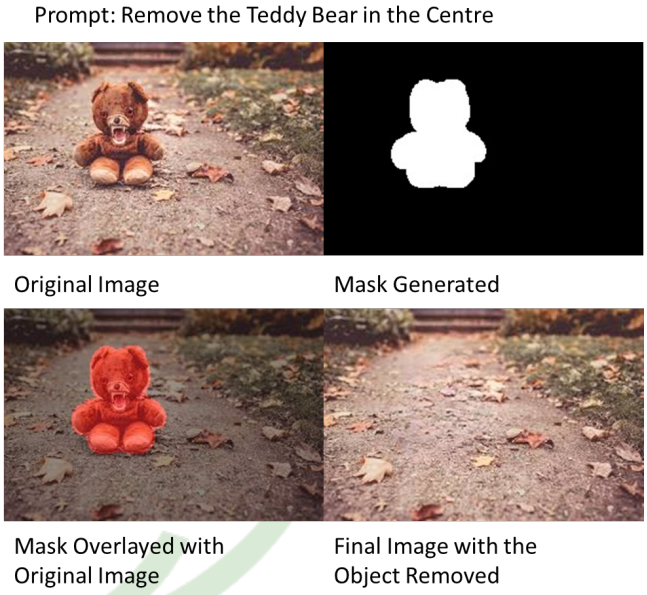

FIGURE 2. Output of Object Removal Pipeline With an Average Latency of 11 seconds

D. TASK 4: SWIPE-GUIDED RELIGHTING

This task enables gesture-driven relighting of an RGB image on mobile devices. A user swipe encodes the desired light direction, which is combined with estimated depth and physically motivated shading priors. A lightweight FiLM-conditioned U-Net (TinyRelightNet) refines this analytic estimate to produce the final relit image. The model is specifically designed to be extremely compact (∼488k parameters) for real-time mobile deployment.

1) Methodology

Swipe-to-Light Mapping. A swipe from $A = (x_1, y_1)$ to $B = (x_2, y_2)$ defines a normalized 2-D direction

$$(\Delta x, \Delta y) = \left(\frac{x_2 - x_1}{W}, \frac{y_2 - y_1}{H} \right),$$

mapped to an initial light vector $(L_x, L_y, L_z) = (2\Delta x, 2\Delta y, z_{\text{offset}})$. Applying user-set intensity I_{user}, we obtain the normalized 3D light vector:

$$L = I_{\text{user}} \frac{(L_x, L_y, L_z)}{\|(L_x, L_y, L_z)\|}.$$

Depth and Normals. MiDaS-small predicts depth D with only a few million parameters, making it suitable for mobile inference. Surface normals are computed via depth gradients:

$$N = \text{normalize}\big(-\nabla D,\, 1\big),$$

yielding geometry-aware cues necessary for shading computation.

Physics-Based Shading Prior. We compute a Lambertian diffuse term:

$$S_{\text{diffuse}} = \max(0,\, N \cdot L),$$

and a rim-light component using the view vector V:

$$S_{\text{rim}} = (1 - \max(0,\, N \cdot V))^{\gamma}.$$

The combined analytic shading is:

$$S_{\text{phys}} = S_{\text{diffuse}} + \lambda_{\text{rim}} S_{\text{rim}},$$

which is then applied multiplicatively to the input image to yield a coarse physically inspired relighting. While informative, this prior lacks texture preservation and may introduce artifacts—thus motivating our refinement network.

FiLM-Conditioned Refinement Network. To refine the analytic estimate, we employ a compact U-Net that operates on the input tensor:

$$X = [I,\, S_{\text{phys}},\, D,\, N] \in \mathbb{R}^{10\times H\times W}.$$

Architecture. TinyRelightNet consists of:

- Encoder with channel widths $[32, 64, 128]$
- Bottleneck at 256 channels
- Decoder mirroring the encoder with skip connections
- Final 1×1 layer predicting a 3-channel residual

Each block uses two 3×3 Conv–ReLU layers. Skip connections preserve high-frequency details essential for texture and edge recovery.

Residual Output. The network predicts a residual R that corrects the physics-based output:

$$\hat{I} = I_{\text{phys}} + R,$$

making the learning task significantly easier and enabling fast convergence.

FiLM Conditioning. Relighting is inherently conditional: the feature activations inside the U-Net must respond to the user-selected lighting direction and intensity. Let

$$F \in \mathbb{R}^{C\times H\times W}$$

denote an intermediate feature map at any U-Net stage, and let

$$L = (L_x, L_y, L_z, I_{\text{user}})$$

be the 4-dimensional light-conditioning vector encoding normalized light direction and user-set intensity.

To modulate features according to illumination, we use Feature-wise Linear Modulation (FiLM), which applies a learned affine transformation to each channel of F:

$$F'_{c,h,w} = \gamma_c(L)\, F_{c,h,w} + \beta_c(L).$$

The scale and shift parameters $(\gamma(L), \beta(L))$ are produced by a lightweight 2-layer MLP:

$$h = \sigma(W_1 L + b_1), \qquad [\gamma(L),\, \beta(L)] = W_2 h + b_2.$$

FiLM therefore allows each feature channel to adapt its behavior based on lighting context: channels associated with highlights may be amplified, shadow-sensitive channels may be suppressed or sharpened, and mid-level features can respond to directional gradients implied by the swipe. Because FiLM operates channel-wise and requires only $O(C)$ parameters, it provides expressive conditional control while preserving the extremely small footprint of TinyRelightNet.

Model Size. The entire TinyRelightNet contains only:

$$488{,}483 \text{ trainable parameters},$$

computed using:

```
sum(p.numel() for p in model.parameters()).
```

This compact design (paired with MiDaS-small's lightweight architecture) enables ONNX FP16/INT8 inference within 10–15 s on consumer mobile devices.

Training Objective. We use a mixed pixel–perceptual loss:

$$\mathcal{L} = \lambda_1 \|\hat{I} - I_{\text{gt}}\|_1 + \lambda_2 \big(1 - \text{SSIM}(\hat{I}, I_{\text{gt}})\big).$$

The ℓ_1 term preserves edges and prevents excessive smoothing, while SSIM promotes perceptual fidelity, local contrast consistency, and structural coherence—particularly important for shadows and highlights in relighting.

Training Setup. Training is performed with AdamW (lr $= 10^{-4}$, weight decay $= 10^{-5}$) and batch size 4, constrained by depth-normal computation and physics shading overhead. Each iteration takes $\sim$ 1 s on a P100 GPU. Thanks to the residual formulation and strong physics prior, convergence is achieved within 5–6 epochs. Validation loss follows training loss closely, indicating good generalization and no overfitting.

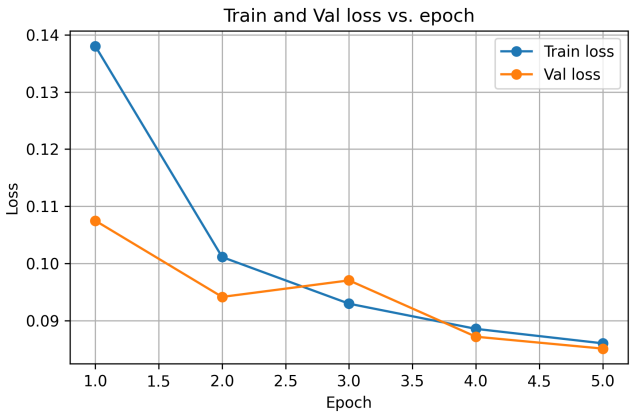

FIGURE 3. Train and Validation Loss vs. Epoch

2) Results

The physics-based shading prior provides globally correct illumination cues, while the FiLM-conditioned TinyRelightNet restores texture, resolves banding, and produces coherent soft shadows and highlights. With MiDaS and the refinement model exported to ONNX and quantized (FP16/INT8), the entire pipeline achieves 10–15 s latency on modern mobile hardware, enabling real-time, gesture-controlled relighting suitable for interactive applications.

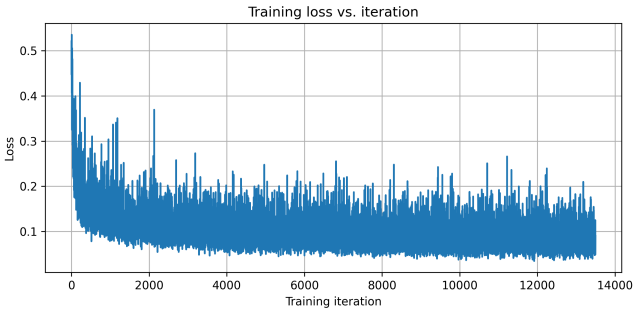

FIGURE 4. Training Loss Vs. Iterations

FIGURE 5. Outputs for Relighting. Arrows Signify the Direction of Swipe by the User.

E. TASK 5: IMAGE REPLACEMENT

The image replacement module enables object-level or background-level regeneration using a diffusion-based inpainting pipeline conditioned on our prompt-guided segmentation system. Depending on the user's instruction, the system either replaces only the selected object or reconstructs the entire background while preserving the foreground. Both workflows rely on the segmentation mask produced earlier, and differ only in whether the mask or its complement is passed to the diffusion model.

1) Methodology

a: Mask Selection.

Let $M^*(x, y) \in \{0, 1\}$ denote the final binary mask obtained from the spatially aware segmentation module. The two replacement modes are defined as:

$$\text{Object replacement: } M_{\text{obj}} = M^*,$$

$$\text{Background replacement: } \mathrm{M_{bg}} = 1 - M^*.$$

These masks specify the region to be regenerated by the diffusion model.

b: Diffusion Backbone.

We adopt Stable Diffusion 1.5 Inpainting as the generative backbone due to its native support for masked inpainting, smaller memory footprint compared to SDXL, and practical CPU performance when combined with efficient sampling strategies. Given the masked image $I \odot (1 - M)$ and prompt P, the model predicts a completed latent representation that is decoded into the inpainted output.

Acceleration via Latent Consistency Models. Standard diffusion sampling requires 20–50 steps, which is prohibitively slow on CPU hardware (approximately 5 minutes per image). To address this, we integrate Latent Consistency Model (LCM) adapters, which reduce sampling to only 5–8 steps:

$$\hat{z}_0 = \mathrm{LCM}\big(z_T, P, M;\ \text{steps} = 5\text{–}8\big).$$

This reduces latency to roughly 15 seconds on typical mobile-class CPUs, enabling real-time object and background replacement.

LoRA Adapters for Quality Enhancement. To improve visual consistency between the generated region and the original image, we apply LoRA adapters specialized for lighting stability, texture coherence, and color harmonization. These adapters integrate directly into the LCM sampling process without increasing computational burden.

2) Results

Object Replacement. We apply the mask M^* to isolate the object region, feed the masked image and diffusion prompt into SD1.5 Inpainting, sample using the LCM scheduler with 5–8 steps, and decode the output. A light edge-aware blending step merges the inpainted object with the untouched background, preserving scene realism.

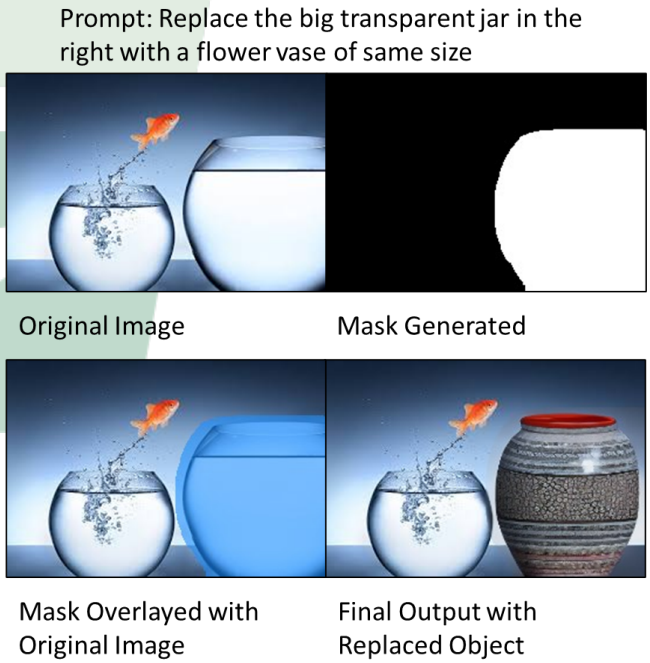

FIGURE 6. Results for Text based Object Replacement with an average latency of 13 seconds

Background Replacement. For background replacement, we use the inverted mask $\tilde{M}(x, y) = 1 - M^*$ and a background-specific prompt. The diffusion model synthesizes a new environment while keeping the foreground intact. The final image is formed by merging the preserved foreground with the generated background.

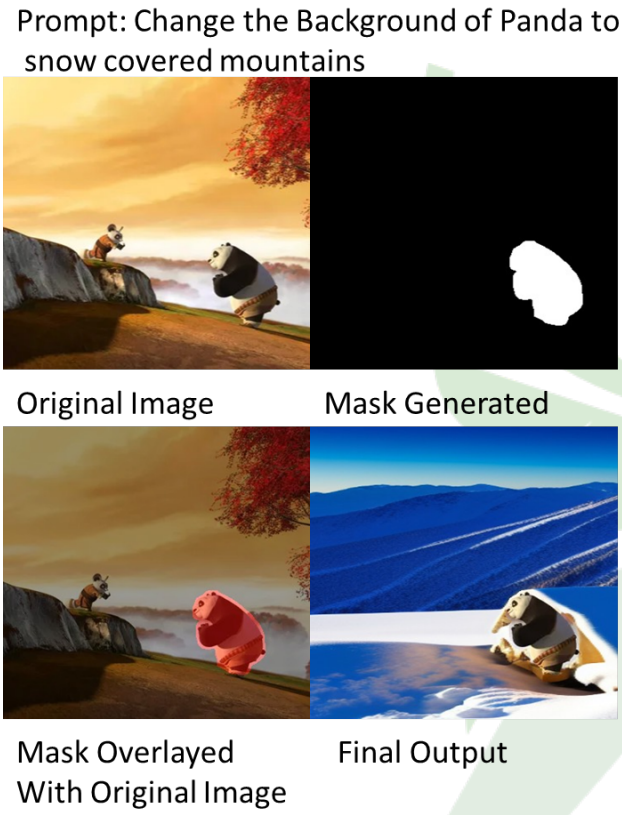

FIGURE 7. Results for Text based Background Replacement with an average latency of 13 seconds

3) Summary

This module performs fully offline generative replacement on CPU-only devices using Stable Diffusion 1.5 Inpainting accelerated with LCM sampling. Object and background replacement are controlled through the mask or its complement, and LoRA adapters ensure stable lighting, texture continuity, and visually coherent blending. With LCM, inference time is reduced from approximately 5 minutes to $\sim$ 15 seconds, making high-quality generative editing feasible on edge hardware.

F. TASK 6: AGENTIC INTEGRATION

The final task in our system focuses on enabling natural, intuitive interaction through an agentic interface. Instead of requiring users to manually select tools or parameters, the system incorporates a lightweight conversational layer that interprets high-level user instructions and automatically maps them to the appropriate visual-editing operations.

1) Methodology

We integrate a compact chatbot built using the LangChain framework with a Mistral-based LLM backend. This agent processes free-form user requests and converts them into structured commands for the editing pipeline. For every input utterance, the agent produces three key fields: (1) `task_type`, specifying the intended operation (e.g., `remove_object`, `relight`, `change_style`); (2) `segmentation_prompt`, a concise phrase used by our prompt-guided segmentation module; and (3) `diffusion_prompt`, a detailed instruction for the downstream inpainting, relighting, or style-transfer model. This structured output serves as the control signal that routes the request to the correct component of the pipeline. The segmentation prompt is forwarded to the text-driven segmentation module, after which the diffusion prompt and task type determine which editing block is executed.

2) Results

This agentic layer significantly enhances usability by enabling a natural-language-driven editing workflow. Users describe desired edits conversationally, and the system automatically handles segmentation, model selection, and parameter configuration. The result is a fluid interaction model that abstracts away technical complexity, reduces user effort, and reliably produces the intended visual transformation without manual intervention.

IV. APP DEVELOPMENT WORKFLOW

The application is implemented using a cross-platform Flutter architecture designed to deliver native-grade rendering, low-latency gesture interaction, and consistent deployment across Android and iOS devices. All core vision models run on-device through ONNX Runtime Mobile, enabling efficient CPU-only execution with ARM NEON acceleration. This setup allows MobileSAM, MobileCLIP, MiDaS-small, LaMa, a refinement U-Net, and SD1.5 to operate within mobile resource constraints while maintaining practical runtimes.

A. ON-DEVICE INFERENCE PIPELINE

All models are exported to ONNX and quantized through a uniform post-training pipeline. FP16 serves as the default precision for general-purpose inference, while INT8 is selectively applied in memory- or latency-critical conditions. This configuration yields usable end-to-end runtimes on edge hardware such as a Pixel 6: MobileSAM executes in 0.9–1.45 s, MobileCLIP performs crop encoding in 0.3 s, LaMa runs in 0.54–0.42 s (FP16/INT8), MiDaS-small processes depth in 0.8 s, and SD1.5 with LCM completes in approximately 15 s. To ensure stability across diverse devices, models are packaged as compressed assets, loaded lazily, unloaded aggressively after use, and supported by tile-based computation for high-resolution inputs.

B. HYBRID CLOUD–EDGE REASONING

The system follows a hybrid architecture that balances mobile constraints with the computational demands of large generative models. Features such as remove, replace, and background-generation rely on server-side inference, where LaMa is used for object removal and Stable Diffusion 1.5 is used for object replacement and background regeneration. Running these models—especially full-resolution diffusion—locally would exceed mobile latency budgets. On the backend, the required models are loaded once at server startup, enabling subsequent inference requests to execute with minimal overhead.

To preserve user privacy in production settings, images processed through the backend are never written to any company-owned database. Instead, they are stored only in a user-specific temporary cache, which may be encrypted to prevent unauthorized access. The frontend performs all vision tasks that can run locally and sends only the necessary masked regions or prompts to the backend when required.

User instructions first pass through a lightweight on-device intent classifier and are then forwarded to a LangChain-managed Mistral API endpoint, which returns a structured JSON describing task type, segmentation cues, diffusion prompts, and spatial hints. Schema enforcement, few-shot prompting, cached responses, and speculative preloading reduce latency and ambiguity, while offline fallbacks allow manual control when cloud access is unavailable.

C. INTERACTION AND UX DESIGN

Interaction design emphasizes responsiveness and real-time feedback. Multi-mode segmentation—including point, box, and sketch inputs—is enabled through precise coordinate transformations feeding into MobileSAM and the MobileCLIP ranking module. Gesture-driven relighting interprets swipe vectors as 3D illumination parameters, leveraging MiDaS depth estimation and a lightweight refinement U-Net. A multi-stage progress pipeline communicates model execution status, while isolated background tasks prevent UI stalls. Consistent user experience is maintained through undo/redo history management and cumulative mask rendering across sketch-based edits.

D. SUMMARY

The resulting system demonstrates a practical mobile deployment of a complex multi-model vision pipeline. Through quantization-aware ONNX deployment, hybrid cloud–edge reasoning, privacy-preserving backend design, and optimized interaction workflows, the application achieves real-time responsiveness for object removal, relighting, segmentation, and text-driven inpainting. Despite its capabilities, the final application bundle remains compact, with a total size of approximately 1.1 GB, making it suitable for production-scale mobile photo editing.

V. CONCLUSION

This work presents a complete end-to-end framework for automated photo editing on edge devices, integrating efficient segmentation, high-quality inpainting, gesture-guided relighting, diffusion-based replacement, and an agentic natural-language interface into a unified deployable system. Using lightweight backbones such as MobileSAM, MobileCLIP, MiDaS-small, quantized LaMa, and LCM-accelerated Stable Diffusion 1.5, the system delivers desktop-level visual fidelity while sustaining practical runtimes on mobile CPUs. A hybrid cloud–edge design enhances scalability, preserves privacy, and maintains low latency for generative tasks.

Across all components, the aim is efficiency without sacrificing quality. Prompt-guided segmentation captures user intent, quantized inpainting restores structure with sub-second latency, swipe-driven relighting provides coherent illumination control, and few-step diffusion enables real-time object or background regeneration. The agentic interface unifies these capabilities by mapping natural-language commands to precise editing operations.

Overall, the system shows that a Photoshop-class editing experience is feasible on resource-constrained devices through optimized vision pipelines. It offers a practical and extensible foundation for next-generation mobile creative tools and moves toward intelligent, on-device photo-editing assistants.

REFERENCES

[1] A. Kirillov *et al.*, "Segment Anything," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2023, pp. 4015–4026. [Online]. Available: https://arxiv.org/abs/2304.02643

[2] C. Zhang *et al.*, "Faster Segment Anything: Towards Lightweight SAM for Mobile Applications," *arXiv preprint arXiv:2306.14289*, 2023. [Online]. Available: https://arxiv.org/abs/2306.14289

[3] P. Vasu *et al.*, "MobileCLIP: Fast Image-Text Models through Multi-Modal Reinforced Training," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2024, pp. 13866–13876. [Online]. Available: https://arxiv.org/abs/2311.17049

[4] R. Suvorov *et al.*, "Resolution-robust Large Mask Inpainting with Fourier Convolutions," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, 2022, pp. 2149–2159. [Online]. Available: https://arxiv.org/abs/2109.07161

[5] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 3, pp. 1623–1637, 2022. [Online]. Available: https://arxiv.org/abs/1907.01341

[6] M. E. Helou *et al.*, "VIDIT: Virtual Image Dataset for Illumination Transfer," *arXiv preprint arXiv:2005.05460*, 2020. [Online]. Available: https://arxiv.org/abs/2005.05460

[7] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-Resolution Image Synthesis with Latent Diffusion Models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2022, pp. 10684–10695. [Online]. Available: https://arxiv.org/abs/2112.10752

[8] S. Luo *et al.*, "Latent Consistency Models: Synthesizing High-Resolution Images with Few-Step Inference," *arXiv preprint arXiv:2310.04378*, 2023. [Online]. Available: https://arxiv.org/abs/2310.04378

[9] E. J. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2022. [Online]. Available: https://arxiv.org/abs/2106.09685

[10] A. Q. Jiang *et al.*, "Mistral 7B," *arXiv preprint arXiv:2310.06825*, 2023. [Online]. Available: https://arxiv.org/abs/2310.06825