REFocus: *Real-world motion-Blur Recovery via a luminance-First conditional Colorization pipeline* — A Two-Stage Approach for Deblurring and Optional Colorization

1. Problem Context and Project Summary

Smartphones, action cams, and surveillance systems frequently capture **motion-blurred, defocused, or noisy** scenes, especially people walking, vehicles in motion, and low-light street shots. Family snapshots also suffer from handshake blur and poor exposure. One-click "enhance" filters often oversharpen or introduce artifacts, while single-stage models struggle to handle real camera blur.
Refocus is a **two-stage pipeline**: **Stage 1 restores structure on luminance** (reduces blur/noise, recovers edges). **Stage 2 colorizes only when the image is grayscale** (e.g., mono CCTV or scanned B&W photos), preserving original luminance/texture. Delivered as a drag-and-drop web tool with Before/After views, Refocus aims to produce **clear, natural images** that are easier to review, search, and share, helping both families and analysts.

2. Dataset

For the two phases, I want to use two datasets. One for restoring the images and one for coloring the images.

2.1. HIDE (Human-Aware Motion Deblurring) from Kaggle - Restoration

- **Type & content:** Paired RGB images with realistic human-centric motion blur (walking people, handheld cameras, street scenes) and corresponding sharp ground truth captured with short exposure. Pairs reflect non-uniform, human-induced blur, closer to surveillance and everyday snapshots than synthetic kernels.
- **Intended use:** Train Stage-1 Restorer in Lab space on L channel only (supervision: L_blur → L_sharp). Decoupling structure (L) from color avoids hue shifts and lets the same model serve both grayscale and color inputs.
- **Scale & splits:** The dataset will be around 16GB with thousands of pairs. We use the official train/val/test lists (or generate a consistent split) and patch-based training at 256×256.

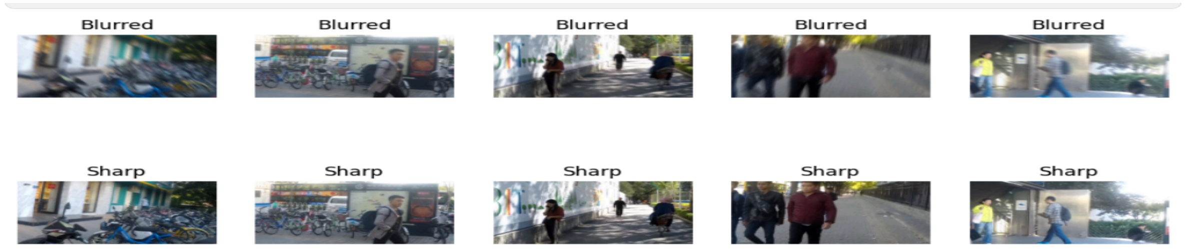

Fig. 1. Dataset

2.2 COCO 2017 (train2017 + val2017), via Hugging Face - Colorization

- **Type & content:** Large, diverse RGB image collection spanning people, objects, and scenes; ideal for learning broad color priors. We treat it as unpaired data for L→ab prediction.
- **Intended use:** Train Stage-2 Colorizer to predict chrominance (ab) from luminance (L). In inference, run colorization only if the restored image is effectively grayscale. Otherwise, preserve original color (ab_original).
- **Scale & splits:** ~118k train2017 images and 5k val2017 images (~18 GB total if fully mirrored). We create an internal validation split from train2017 (≈95/5) and use val2017 as the test set for final reporting.

3. Preprocessing

Stage 1: Restoration (HIDE)

- Each training example has two images: a blurry photo and its sharp version.
- Then, converted both to Lab color and used only the L channel (lightness/brightness).
- scaled L to 0–1 and clip values so nothing is outside the range.
- Train: From the input image, then center-crop to 256×256.
- Apply the same crop to blurry and sharp, and sometimes flip left/right.
- Val/Test: just center-crop to 256×256.
- From HIDE's training images, we keep about 90% for training and 10% for validation.
- We use HIDE's test set as the final check.

Stage 2: Colorization (COCO via Hugging Face)

- We download images on the fly from official COCO URLs (and cache them on disk).
- Convert each image to Lab. Input to the model: L (lightness) in 0–1
- Target to learn: ab (color) in −1 to 1
- Clipped after conversion to keep values in range.
- Train: From the input, center-crop to 256×256, sometimes flip.
- Val/Test: center-crop to 256×256, no flip.
- COCO train2017 (~112k images) → we make our own split:
- 95% train (106,655 images) and 5% validation (5,613 images).
- COCO val2017 (5,000 images) is our test set.
- Many images have little color. To stop the model from becoming "too gray," we give more weight to pixels that actually have strong color when computing the loss.

Ethical / Privacy considerations

- state how long uploads are kept (or don't store them at all); provide a "Delete after download" option.

- use images only for research; document the source; avoid faces in public demos without consent.

4. Planned Architecture

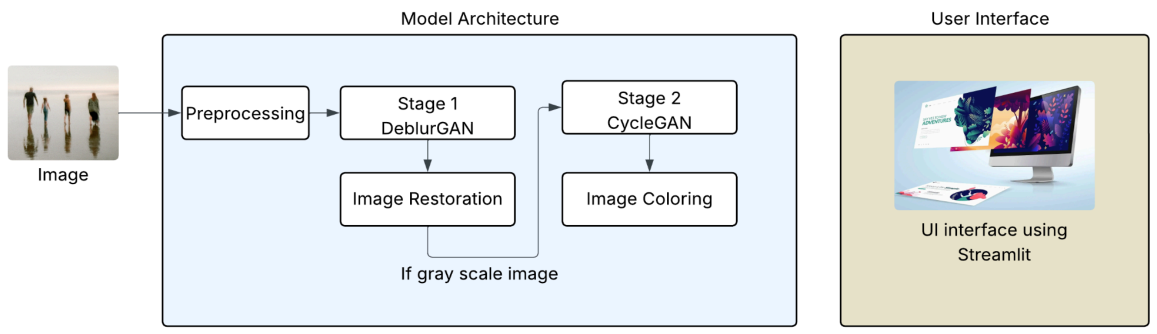

Fig.2. Architecture

1. Input will be Image RGB

2. Preprocess (convert to brightness + color layers, resize/crop, clamp)

3. Stage-1: DeblurGAN (Restoration on brightness only)

- Generator restores the brightness of the image (blurry → sharp)
- Discriminator checks the realism of the restored brightness
- Output: restored brightness image

"Is it grayscale?" check

- If the original has little/no color → go to Stage-2
- Else keep the original color layers

4. Stage-2: CycleGAN (Colorization)

- Domain X: grayscale (brightness duplicated to 3 channels)
- Domain Y: natural color photos (from COCO)
- Generator X→Y: gray → color (predicts color layers or full RGB)
- Generator Y→X: color → gray
- Cycle + identity losses keep hues stable and realistic
- Output: predicted color layers (or a color image you can convert)

5. Recompose: combine restored brightness + chosen color layers → RGB → Streamlit UI showing both the blurred and sharpened images.

5. User Interface Plan

- I want to build the interface in **Streamlit** because it supports rapid prototyping, easy side-by-side image display, and simple deployment. The app will run locally during development and can be hosted on Streamlit Community Cloud for demos.
- The interface will accept a single image upload in JPG, PNG, or TIFF format. In the sidebar, users will choose a processing mode like “Restore only,” “Restore + Colorize,” or “Always colorize”. They can toggle “Auto-detect grayscale” to run colorization only when the input has little or no color.
- The app will display a **side-by-side** comparison. The original image on the left and the processed image on the right, each scaled to fit the page and usable with the standard Streamlit zoom behavior.

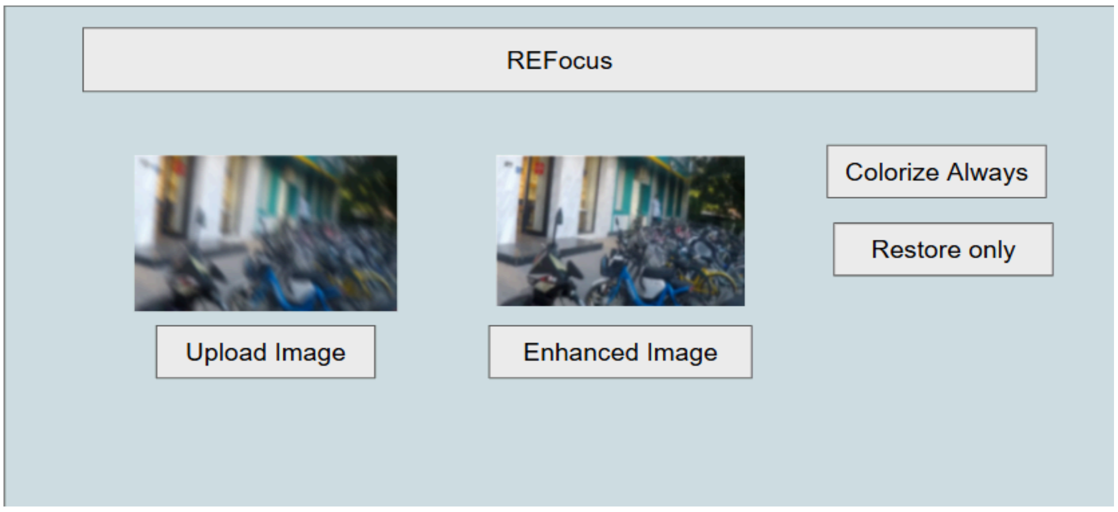

Fig. 3. Sample UI

6. Innovation and Anticipated Challenges

- **Domain gap (real scenes vs. training data).**

  Risk: Low light, compression, or fine artifacts may be under-represented.

  Mitigation: Add input-only augmentations (vignetting, JPEG noise, light scratches), run early checks on a small curated set, and, if needed, fine-tune on a few real examples.

- **GAN stability and over/under-restoration.**

  Risk: Adversarial training can ring or hallucinate. Pure content loss can oversmooth.

  Mitigation: Warm-start DeblurGAN with content/SSIM before enabling GAN loss. Use PatchGAN with instance/spectral norm. Monitor PSNR/SSIM and visual grids. Expose a “deblur strength” slider to blend results in the UI.

- **Desaturation or hue drift in colorization.**

  Risk: The model may produce dull colors or incorrect tints.

  Mitigation: Use a **color-aware (chroma-weighted) loss**, keep **identity loss** in CycleGAN, and always **replace the output's brightness with the restored brightness** before rendering. Provide a "restore-only" option and trigger colorization only when grayscale is detected.

7. Implementation Timeline

Week	Focus	Example
Oct 20 - 26	Set up HIDE/COCO loaders, preprocessing, fixed splits; scaffold DeblurGAN (restore) + CycleGAN (color).	Data ready; baseline scripts run.
Oct 27 – Nov 2	Train DeblurGAN (warm-start with content/SSIM, then GAN); add val PSNR/SSIM.	Stable restorer + checkpoints.
Nov 3 – 16	Train CycleGAN (gray↔color) with identity + cycle + color-aware loss; compose restored brightness + color.	Plausible colorization; compose working.
Nov 17 – 30	Build Streamlit UI: upload, modes, auto-gray, strength slider, face-blur, EXIF strip, side-by-side view. Refinement	Working interactive demo. Refinement
Dec 1 – 11	Rehearse demo. Finalize report, GitHub, and slides.	Presentation-ready deliverables.

Table.1. Timeline

8. Responsible AI Reflection

- Fairness & Representation. Colorization can shift skin tones or cultural artifacts. I will include diverse examples in validation, review failures, and keep conservative defaults. A "Restore only" mode lets users avoid AI-generated color when it's inappropriate.
- Privacy & Consent. Surveillance-like images may contain bystanders. The app will offer face-blur and EXIF stripping by default in demos. I will avoid redistributing input images and process files in memory or delete them after download.
- Data Licensing & Provenance. I will respect COCO/HIDE licenses and keep links/attribution. I will not re-share raw datasets; I will share only derived results where allowed and record the dataset/source in experiment logs.