Bird Feeder Image Editing

Prepared by – Aakash Maskara

Overview

This project focuses on leveraging YOLOv8 segmentation models and Stable Diffusion pipelines to perform object-centric transformations on bird-feeder images. The primary objectives included:

- Selecting images with birds
- Segmenting bird masks
- Removing birds via background inpainting
- Replacing birds with different birds
- Replacing birds with squirrels or chipmunks
- Generating completely synthetic images showing both birds and squirrels at birdfeeders using text-to-image models

All processing has been done in Python using Ultralytics, diffusers, torch, and PIL.

Dataset Insights

The dataset consisted of 20 natural JPG images, among which only a subset showed both a bird and a birdfeeder. The pipeline had to filter these images automatically.

- Input Images : 20 original JPG images (origIm1.jpg through origIm20.jpg)
- Relevant Images : 5 Images with both a bird and birdfeeder were selected for this project
- Masks & Outputs : Segmentation masks were generated for selected images, and corresponding altered images were created for three inpainting tasks

Training and Optimization

- No neural network was trained from scratch

- All modules use pre-trained detection or generative models
- Optimization focused on prompt engineering, bounding box padding, and inpainting model selection.

Section-wise Implementation and Analysis

1. subSelectImages.py – Image Subselection

   Objective : Identify and select 5 out of 20 images (origIm1.jpg to origIm20.jpg) containing birds with high confidence.

   Model Used : YOLOv8s object detection model (yolov8s.pt).

   Logic :

   - Threshold: Confidence > 0.8
   - Bird Class ID : 14
   - First 5 images satisfying the bird detection criterion were selected

   Implementation Summary :

   - Reads all 20 images
   - Uses ultralytics to detect birds
   - Prints file names of images selected

2. segmentImages.py – Bird Segmentation

   Objective : Segment bird masks from the 20 original images.

   Model Used : YOLOv8x-seg segmentation model (yolov8x-seg.pt).

   Output : Binary masks saved as origImX_mask.png in the same directory.

   Mask Processing Steps :

   - Class ID check and confidence thresholding (CONF_THRESHOLD = 0.7)
   - Bitwise OR operation for combining multiple bird masks in an image

- Masks resized to original image dimensions

3. removeBirds.py – Inpainting to Remove Birds

   Objective : Remove birds from the selected 5 images using inpainting and fill the removed region with natural background.

   Model Used : StableDiffusionInpaintPipeline (stabilityai/stable-diffusion-inpainting).

   Pipeline : Loaded on CPU using torch_dtype=torch.float32

   Prompt : "natural background with trees and bird feeder, no birds"

   Key Features :

   - Inpainting based only on mask and surrounding image
   - Output saved as origImX-birdsRemoved.jpg

4. replaceBirds.py – Replace Birds with Other Birds

   Objective : Replace the birds in the selected images with different bird species using generative inpainting.

   Model Used : StableDiffusionInpaintPipeline (stabilityai/stable-diffusion-inpainting)

   Pipeline : CPU based, with rescaled images to 512x512.

   Output : Images saved as origImX-birdsReplaced.jpg.

5. substituteSquirrels.py – Replace Birds with Squirrels

   Objective : Replace birds with squirrels or chipmunks in the same location.

   Model Used : Same inpainting model as replaceBirds.py

   Prompt : "a realistic squirrel or chipmunk sitting on a tree branch or near a bird feeder"

   Output : Files saved as origImX-NowWithSquirrels.jpg

   Logic : Used the same 5 selected images and masks, keeping model structure consistent for fair comparison.

6. generateBirdFeederImagesFromText.py – Text-to-Image Generation

   Objective : Generate synthetic images of birds and squirrels interacting at bird feeders using text-to-image pipeline.

   Model Used : StableDiffusionPipeline (stabilityai/sd-turbo)

   Prompt : "a realistic backyard scene showing squirrels and birds eating seeds together at a bird feeder, natural lighting, photo taken with a DSLR camera"

   Negative Prompt : "blurry, low resolution, distorted, text, watermark, double heads, unnatural colors"

   Generation Details :

   - 5 images generated with seed increment
   - num_inference_steps=4, guidance_scale=1.0
   - Saved as generatedImage_1.png to generatedImage_5.png

Experiments Conducted Before Reaching Final Model

File	Experiment	Outcome
subSelectImages.py	Tried YOLOv8x initially, slow for detection, switched to YOLOv8s for faster runtime.	Achieved 5 confident bird detections.

segmentImages.py	Used CONF_THRESHOLD=0.5 first, but it introduced false positives. Adjusted to 0.7 for reliability.	Produced clean bird masks.
removeBirds.py	Initially used default prompt, birds were replaced with other animals. Fixed by using explicit "no birds" phrasing.	Final output resembled original backgrounds without birds.
replaceBirds.py	Multiple prompts were tested: "parrot", "bird species" etc., but caused unnatural merges. Finalized prompt with broader "exotic colorful birds".	Output images generated new birds of different types.
substituteSquirrels.py	First prompt resulted in squirrels alone. Final prompt described chipmunks/squirrels near feeders.	Improved results but some ambiguity remained.
generateBirdFeederImagesFromText.py	Tried multiple models (realisticvision, sd-v1-5), poor compatibility. Best results with sd-turbo. Prompt tuned iteratively.	Final prompt gave best balance of realism and object clarity.

Final Results

Birds Removed –

Birds Replaced –

Squirrels Substituted –

Generated From Text –

Peer Model Comparison

To benchmark my model performance, I compared results with a peer – Julian de Nijs (jd3846@drexel.edu) :

1. subSelectImages.py :

Aspect	My Model	Julian's Model
Model Used	YOLOv8s from Ultralytics	YOLOv8s from Ultralytics
Thresholds & Selection	Used bird and bird feeder as required classes, filters confidently	Similar filtering but lacks precise bounding box union logic
Output Handling	Combined both detections and computed union BBox	Focused more on individual classes, no explicit overlap logic

2. segmentImages.py :

Aspect	My Model	Julian's Model
Segmentation Model	YOLOv8 segmentation model	YOLOv8 segmentation model
Confidence Threshold	Tuned to 0.7 for reducing false positives	Used 0.5, leading to more false segmentations
Post-processing	Applied binary masks per detected bird ID	Directly saved raw mask outputs, less refined

3. removeBirds.py :

Aspect	My Model	Julian’s Model
Inpainting Model	stabilityai/stable-diffusion-2-inpainting	stabilityai/stable-diffusion-2-inpainting
Prompt Engineering	"background, no birds, empty scene" to avoid unwanted animal inpainting	Used default "natural background" leading to inconsistent results
Image Preprocessing	Resized to 512x512, converted to RGB, mask to black	Used same, but no explicit resizing fallback

4. replaceBirds.py

Aspect	My Model	Julian’s Model
Prompt	"exotic colorful bird" (broad, vivid)	"a parrot" (too narrow repetition)
Seed Control	Random seed set	seed set to 0 for all resulting in less variety
Visual Diversity	images include different bird types	Output has repeated parrots in similar poses

5. substituteSquirrels.py

Aspect	My Model	Julian’s Model
Prompt Design	"realistic squirrel or chipmunk sitting near a bird feeder"	"a squirrel" only
Prompt Structure	Considers location context (on branch or near feeder)	No context, leading to floating/unrealistic placement
Results	Better squirrel placement in context-rich backgrounds	Results float mid-air or overlap artifacts

6. generateBirdFeederImagesFromText.py

Aspect	My Model	Julian’s Model
Prompt	"squirrels and birds eating seeds at feeder, DSLR, natural"	"bird feeder with squirrels and birds"
Negative Prompt	Actively removes noise: "blurry, distorted, watermark, etc."	Does not use negative prompting
Image Quality	Clearer, consistent object placement, balanced saturation	Images sometimes include blurred shapes and watermarks

Conclusion

This modular pipeline successfully executes the required tasks in a robust and scalable fashion. Compared to peer implementations, better image realism and flexibility were achieved through careful tuning of prompts, bounding box padding, and mask handling. The segmentation and inpainting synergy played a critical role in realistic outputs.