GloPER: Unsupervised Animal Pattern Extraction from Local Reconstruction (Supplementary)

Bowen Chen Yun Sing Koh Gillian Dobbie
The University of Auckland, New Zealand
bche264@aucklanduni.ac.nz y.koh@auckland.ac.nz g.dobbie@auckland.ac.nz

Reproducibility

All our experiments were conducted on two NVIDIA RTX 6000 GPUs with 64GB of RAM. The framework is implemented in Python 3.12.5 with the following key packages and their corresponding dependencies:
- rembg 2.0.64
- pytorch 2.6.0+cu126
- cv2 4.11.0
- numpy 2.1.3
- scipy 1.15.1
- PIL 11.1.0
- torchvision 0.21.0+cu126
- transformer 4.48.2
- sklearn 1.6.1
- seaborn 0.13.2

Dataset Labeling Process

The labeling was conducted using Clip Studio Paint (CSP), a digital illustration software that provides a wide range of annotation tools. The annotator had full access to CSP's features, including layering, fill tools, and fine-tuning brushes (e.g., leftover pen) to ensure precise segmentation of patterns. The primary goal of the labeling process was to capture meaningful patterns while disregarding background elements. The annotator was instructed to prioritize visually distinctive patterns, ensuring that the labeled regions align with natural texture variations and structural details. No predefined criteria were imposed regarding what constitutes an "important pattern," allowing the annotator to use their domain knowledge and perceptual judgment to determine which markings should be highlighted.

To maintain consistency and quality, the annotator: (1) Used multiple layers to separate different patterns before merging final labels. (2) Adjusted brush settings to refine complex or fine-grained textures. (3) Reviewed and iterated on labels to minimize subjectivity and ensure coherence across different images. The labeling process aimed to create high-quality, perceptually guided annotations that emphasize patterns critical for segmentation, making the dataset well-suited for fine-grained pattern extraction and recognition tasks .

Baseline Implementation Details

K-Means Clustering.

K-Means [1, 6] clustering segments images based on color similarity, grouping pixels into k clusters. To improve robustness, we apply K-Means only to foreground pixels, using a background mask from the rembg library to exclude irrelevant regions. The clustering is performed in the RGB color space using OpenCV's implementation with 20 iterations and a random initialization strategy. The resulting pixel labels are then reshaped back to the image space, forming k binary segmentation masks. For pattern dice score calculation, given that k-means clustering does not know which colors correspond to the animal pattern, both the segmentation output and the inverse segmentation output are computed, taking the better one.

Watershed Segmentation.

The Watershed algorithm [2] performs segmentation by identifying image gradients and intensity variations. The image is first converted to grayscale and smoothed using Gaussian blur to reduce noise. We then apply Otsu's thresholding to separate foreground and background, followed by morphological operations to refine segmentation boundaries. Markers are assigned to connected components, and unknown regions are labeled as zero. The Watershed algorithm then propagates segmentation boundaries based on gradient flows.

Segment Anything Model (SAM).

SAM [4] generates a set of segmentation masks given by the $mask\_generator$ predefined function. To adapt SAM for fine-grained pattern segmentation, we rank each segmentation mask by area to extract the largest and second-largest regions. We then construct a complementary binary mask: one containing the largest identified pattern and another including the remaining segmentations. For our implementa-

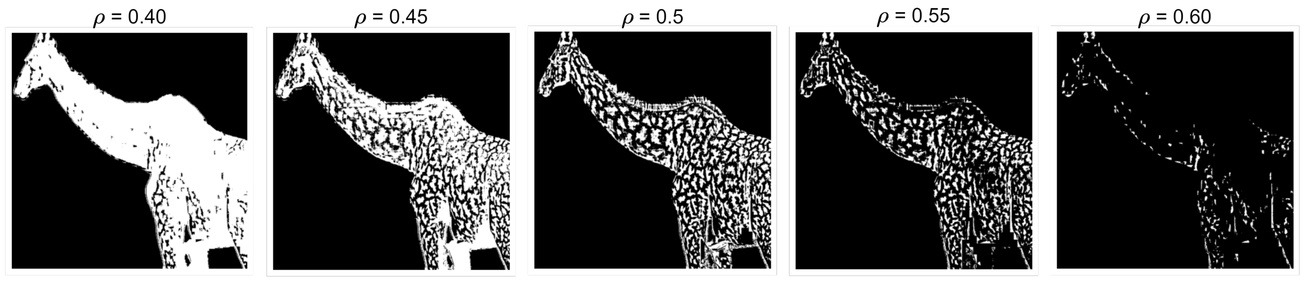

Figure 1. **Effect of threshold ρ on Segmentation Output.** A low threshold results in increased false positive pattern outputs, and a higher threshold results in reduced true positive pattern outputs.

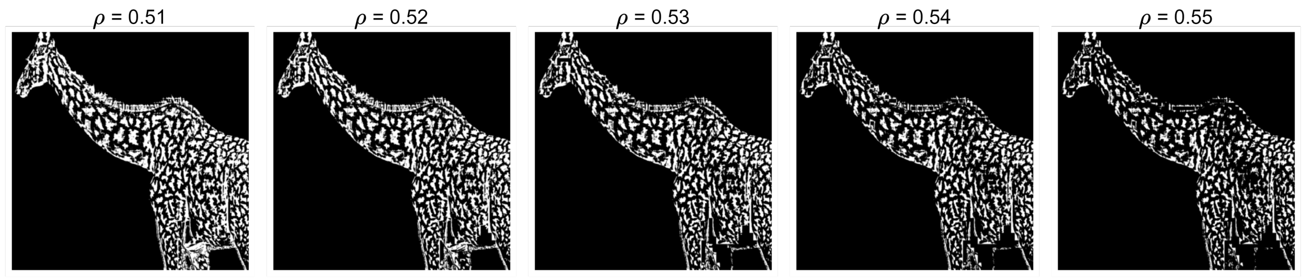

Figure 2. **Effect of Small Threshold ρ Increments on Segmentation Output.** As the threshold increases, more pattern details are lost, leaving only the most confident patterns.

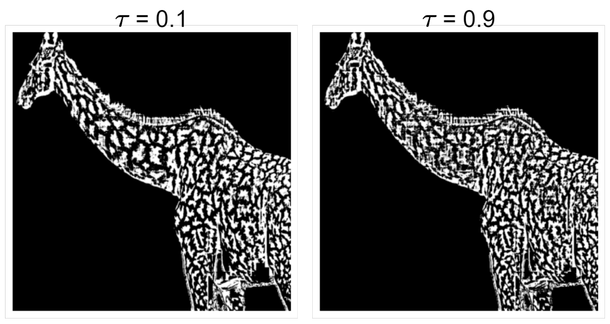

Figure 3. **Effect of temperature τ on Segmentation Output.** A low temperature results in sharper segmentation outputs.

tion, we use SAM's ViT-H model, loaded from the checkpoint $sam\_vit\_h\_4b8939.pth$. The mask generator is initialized with 64 points per side and a two-layer cropping for finer mask quality.

CLIP-Seg.

CLIP-Seg [5] segments images based on text prompts, aligning text and image embeddings through CLIP's vision-language model. Given an input image and a textual description (e.g., "pattern of animal"), CLIP-Seg generates a probability map, which is then thresholded to obtain a binary segmentation mask. We implement CLIP-Seg using the pretrained $clipseg-rd64-refined$ model from Huggingface transformer library, along with its corresponding preprocessor.

DINO.

DINO [3, 7] performs self-supervised feature extraction to generate high-level image representations. The model processes the image through multiple transformer layers, producing a feature map that is compressed using PCA into three dimensions corresponding to RGB. The PCA-reduced features are converted to grayscale, thresholded at 0.5 to obtain binary masks, and refined using a background mask to remove irrelevant regions. We use Facebook's pretrained $dinov2\_vitg14\_reg$ model, loaded from Torch Hub.

Hyperparameter Ablation Studies

To analyze the impact of key hyperparameters on segmentation performance, we conduct ablation studies on the temperature parameter (τ) and the threshold parameter (ρ).

Effects of Temperature.

The temperature parameter (τ) in the Gumbel-Softmax function controls the degree of discretization in the soft assignments. When τ is high, the output distribution is soft and continuous, leading to smoother probability maps.

When τ is low, the function closely approximates a discrete one-hot assignment, resulting in sharper segmentations. However, since the final segmentation mask is obtained via a separate thresholding step rather than directly using the raw Gumbel-Softmax output, τ has a relatively minor influence on the final binary segmentation mask. As shown in Figure 3, lowering τ slightly increases segmentation sharpness but does not dramatically alter the final output.

Effects of Threshold.

The threshold parameter (ρ) is applied after segmentation probabilities are generated, determining the final binary mask. It controls the strictness of pattern selection, with higher values favoring a more conservative segmentation that only retains pixels with high confidence. As shown in Figure 1 and Figure 2: Low ρ (lenient threshold) leads to more false positives, capturing more pixels as patterns but also introducing noise. High ρ (strict threshold) results in reduced true positives, filtering out weakly confident pixels and potentially missing true patterns. While lower thresholds may allow the model to capture more subtle textures, they also increase the risk of including background noise. On the other hand, higher thresholds enhance segmentation precision but can cause over-segmentation, where only the most confident pixels are retained, discarding faint but valid patterns.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11): 2274–2282, 2012. 1

[2] Serge Beucher and Fernand Meyer. The morphological approach to segmentation: the watershed transformation. In *Mathematical morphology in image processing*, pages 433–481. CRC Press, 2018. 1

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 2

[4] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023. 1

[5] Timo Lüddecke and Alexander Ecker. Image segmentation using text and image prompts. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7086–7096, 2022. 2

[6] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–298. University of California press, 1967. 1

[7] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 2