DSG BYOP Reproducibility Track

**Reproducibility report: Learning Hierarchical Image Segmentation For Recognition and By Recognition**

By Jheel Maheshwari

INTRODUCTION

The paper introduces a framework aimed at addressing the challenge of hierarchical image segmentation through a process of recognition-driven segmentation. Traditional image recognition and segmentation models often treat these tasks independently, leading to inefficiencies and limited mutual reinforcement. This work proposes a paradigm shift by embedding segmentation as an inherent part of the recognition process, unifying the two into a single, cohesive framework.

Key to this approach is the use of **Concurrent Adaptive Segment Tokens (CAST)**, a model designed to progressively group image regions into hierarchical structures. Instead of relying on predefined square patches, the model utilizes superpixels that adapt to image contours, enhancing the alignment between segmentation and recognition tasks. The framework achieves this by employing **graph pooling**, which aggregates smaller segments into larger, semantically meaningful groups in a structured hierarchy, facilitating both fine-grained and holistic recognition.

Unlike traditional methods requiring explicit supervision for segmentation, the model learns segmentation implicitly through recognition objectives. This eliminates the need for standalone segmentation datasets and training, making the framework both efficient and scalable.

The paper highlights two key pillars of the proposed approach:

1. **Recognition-Driven Hierarchical Segmentation**:  
   The segmentation process is guided by recognition tasks, ensuring that the hierarchical structures formed are optimized for improving recognition performance. By unifying segmentation and recognition, the framework eliminates redundancies and creates a seamless flow of information.
2. **Adaptive Graph Pooling**:  
   Leveraging graph pooling techniques, the model adaptively groups superpixels into meaningful hierarchies. This enables the segmentation process to dynamically adjust to different image structures and recognition challenges, ensuring that the model remains robust across varying tasks and datasets.

The authors address the limitations of traditional patch-based methods and demonstrate how CAST outperforms state-of-the-art models like Vision Transformers (ViT) and SAM on unsupervised segmentation and recognition tasks, achieving superior accuracy and efficiency. This framework represents a significant advancement in the unification of segmentation and recognition for vision models.

Scope of Reproducibility  
  
1) The paper claims that CAST eliminates the need for explicit supervision in segmentation by learning segmentation as part of the recognition task, significantly improving efficiency.

2) The use of superpixels as adaptive image regions improves segmentation quality by aligning segmentation with natural image contours better than traditional patch-based approaches.

3) Graph pooling allows for progressive grouping of image regions into a meaningful hierarchical structure, enhancing both fine-grained and holistic recognition tasks.

4) CAST achieves good results on unsupervised segmentation and recognition tasks, outperforming baseline models like ViT and SAM.

5) The hierarchical segmentation and recognition integration reduces computational overhead, enabling better performance on smaller datasets.

This reproducibility study will aim to examine the following claims and experiments:

* Validate the integration of segmentation and recognition tasks and its impact on accuracy and efficiency.
* Replicate the use of superpixels and analyse their contribution to segmentation quality.
* Assess the role of graph pooling in creating meaningful hierarchical groupings.
* Compare the performance of CAST with traditional models (ViT, SAM) across standard benchmarks.
* Evaluate the computational efficiency and scalability of CAST on smaller datasets.
* Experiment with different configurations of graph pooling and superpixel parameters to understand their effects on model performance.

This study will primarily focus on implementing the CAST model and its hierarchical segmentation mechanism to reproduce the above claims and analyse their feasibility and generalizability.

METHODOLOGY

The implementation of the paper relied on the authors' provided codebase, which was mostly intact and usable for reproduction. Using their repository as the foundation, I focused on understanding, fine-tuning, and adapting the algorithm for experimental and analytical purposes.

I utilized PyTorch as the primary framework for executing the code, supported by extensive documentation and resources to clarify ambiguities. I opted for Google Colab environment with T4 GPU accelerators, as it provided sufficient resources for multiple iterations.

The code was primarily used to reproduce the CAST framework described in the paper. I extended its implementation to cover hierarchical segmentation with adaptive superpixels and progressive graph pooling. Adjustments were made to hyperparameters and specific components of the code to evaluate their impact on segmentation and recognition tasks.

The main task included testing the model on PartImageNet, focusing on hierarchical image segmentation and part-to-whole recognition. I successfully replicated the results for unsupervised hierarchical segmentation and extended the evaluation to include comparisons with ViT and SAM.

**Ablation Studies:**  
To comprehensively evaluate the performance and adaptability of the algorithm, I conducted ablation studies by experimenting with the critical parameters:

* Trying a different algorithm for making superpixels: The authors have used SEEDS Algorithm, I used SLIC instead and reported results.
* Experimenting with the number of segments and observing how increasing the segments affect the performance: 196 segments have been used originally, I tried 256, 384, 512, 768 and 1024.
* Trying simpler graph pooling technique: The Graph Pooling Technique used uses attention mechanism. I attempted to try a simpler graph pooling technique, where each node in the next layer is formed by averaging the features of connected nodes in the current layer.
* Trying a dynamic graph pooling technique which dynamically groups nodes into clusters based on learned or feature-driven similarity metrics.

These experiments underscored the robustness of the original implementation and provided insights into possible extensions for future work, such as applying CAST to larger datasets like ImageNet or enhancing its segmentation granularity.

MODELS

CAST (Segmenter for Recognition): Its framework modifies the Vision Transformer (ViT) by integrating adaptive superpixels and graph pooling, creating a hierarchical segmentation structure directly tied to recognition. CAST performs segmentation and recognition simultaneously, requiring only image-level supervision. It excels in both segmentation and recognition tasks, outperforming SAM, HSG, and ViT. However, it demands careful parameter tuning to achieve optimal efficiency.

SAM (Segment Anything Model): SAM is a boundary-focused segmentation model trained on 11 million images and 1 billion masks. It excels at delineating object boundaries across multiple granularities and produces high-quality segmentations. However, it lacks hierarchical consistency and is not integrated with recognition tasks. In this study, SAM was used as a benchmark against CAST for segmentation tasks on datasets like PartImageNet and DensePose.

HSG (Hierarchical Semantic Grouping): This model specializes in hierarchical segmentation by producing consistent segmentations across multiple granularities using a ResNet-based architecture. It is highly effective for segmentation tasks but lacks integration with recognition. HSG was benchmarked against CAST on tasks like PartImageNet and DensePose to evaluate segmentation performance.

ViT (Vision Transformer): It processes images by dividing them into fixed-size patches (e.g., 16x16) and using self-attention for classification. It delivers strong classification performance, especially on large datasets. However, ViT lacks built-in segmentation capabilities, and additional clustering steps are required for segmentation tasks. In this study, ViT tokens were clustered for hierarchical segmentation and compared to CAST’s segmentation capabilities.

An implementation on Swin-T was also considered in this study for its hierarchical architecture, but full implementation faced execution challenges due to compatibility issues.

A description of the Swin Transformer model is provided below:

* **Hierarchical Design:** Creates multi-resolution feature maps by progressively merging window features, enabling effective multi-scale representation.
* **Shifted Windows:** Processes images in non-overlapping windows that shift between layers, enabling cross-window connections while reducing computational complexity.
* **Window-Based Self-Attention:** Computes self-attention within individual windows, reducing the quadratic complexity of traditional Transformers to linear levels relative to image size.

DATASETS

Main datasets used for training:

1) ImageNet

* Total number of examples: ~1.2 million labeled images (training).
* Number of classes: 1,000 object categories.
* Class labels: Includes animals, vehicles, instruments, and more.
* Image size: Variable, often resized to 224x224 pixels for training.

2) COCO (Common Objects in Context)

* Total number of examples: ~330,000 images (118,000 training, 5,000 validation, and ~40,000 testing).
* Number of classes: 80 object categories.
* Class labels: People, animals, vehicles, household objects, outdoor objects, etc.
* Image size: Variable, typically resized to 640x640 pixels for detection tasks.

Main dataset used for evaluation:

PartImageNet\_OOD

* Total number of images: ~24000 (16540 training, 2957 validation, and 4598 testing)
* Number of classes: 11 object categories, further annotated into 40 part categories.
* Class labels: Objects such as "Biped," "Fish," "Quadruped," with parts like "Head," "Torso," and "Legs."
* Image size: Variable, aligned with original ImageNet images.

Other datasets aimed to be used for evaluation:

DensePose

* Total number of examples: ~50,000 annotated images.
* Number of classes: Human body parts, divided into 14 categories.
* Class labels: Head, torso, upper limbs, lower limbs, etc.
* Image size: Variable; annotations are dense UV maps over human figures.

PASCAL VOC

* Total number of examples: ~21,000 images
* Number of classes: 20 object categories.
* Class labels: Animals, vehicles, household objects, and outdoor objects.
* Image size: Variable, typically resized to 512x512 pixels for training.

PASCAL Context

* Total number of examples: ~10,000 images with extensive contextual labels.
* Number of classes: Over 400 annotated categories.
* Class labels: Includes objects, parts, and background categories such as “sky,” “ground,” and “table.”
* Image size: Variable, aligned with original PASCAL VOC dataset.

ADE20K

* Total number of examples: ~27,000 images (22,000 training, 2,000 validation, and 3,000 testing).
* Number of classes: 150 semantic categories.
* Class labels: Includes indoor and outdoor scenes, e.g., walls, doors, people, furniture, streets.
* Image size: Variable, typically resized to 512x512 pixels for training.

Preprocessing Steps done for evaluation on PartImageNet:

1. **Resizing:** The images were resized to 224x224 pixels to match the input size expected by the model.
2. **Normalization:** Images were normalized using the mean and standard deviation of the ImageNet dataset: mean=[0.485, 0.456, 0.406] and std=[0.229, 0.224, 0.225].
3. **Center Crop:** A 224x224 pixel square from the center of an image after resizing, ensuring standardized input for models while focusing on the central content.
4. **ToTensor Conversion:** Images were converted to PyTorch tensors to facilitate computations on the GPU.
5. **Superpixel generation:** Superpixels were generated using SEEDS algorithm.

Other datasets have also gone through same preprocessing steps. Evaluation on all the datasets couldn’t be done in time due to data structure inconsistency and time constraints.

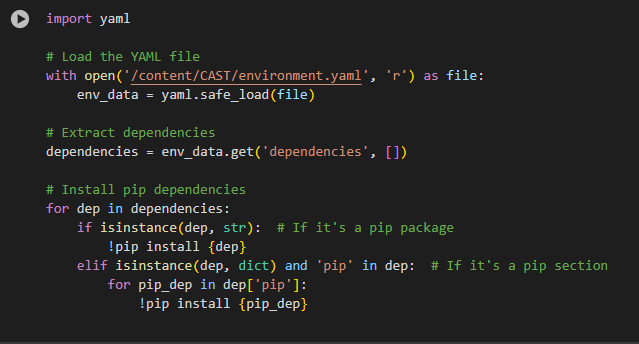
EXPERIMENTAL SETUP

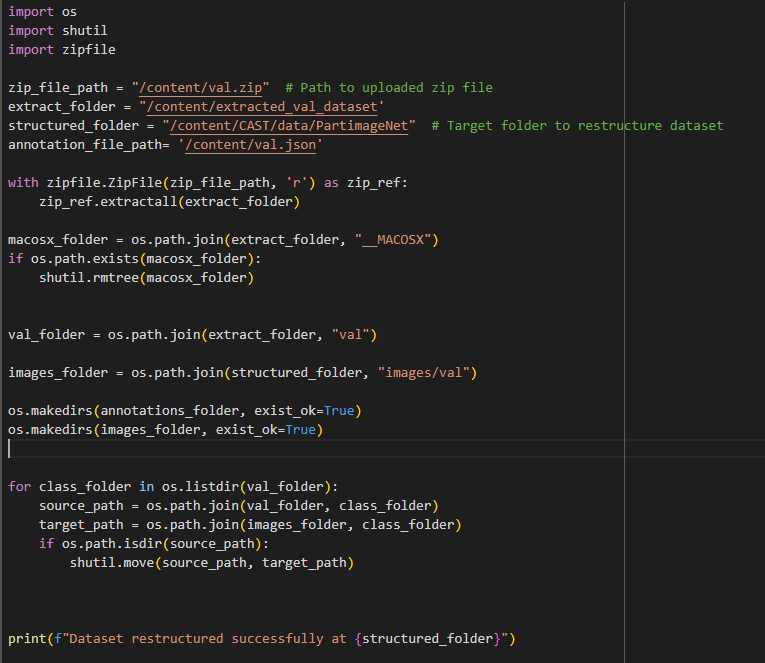
Environment setup:

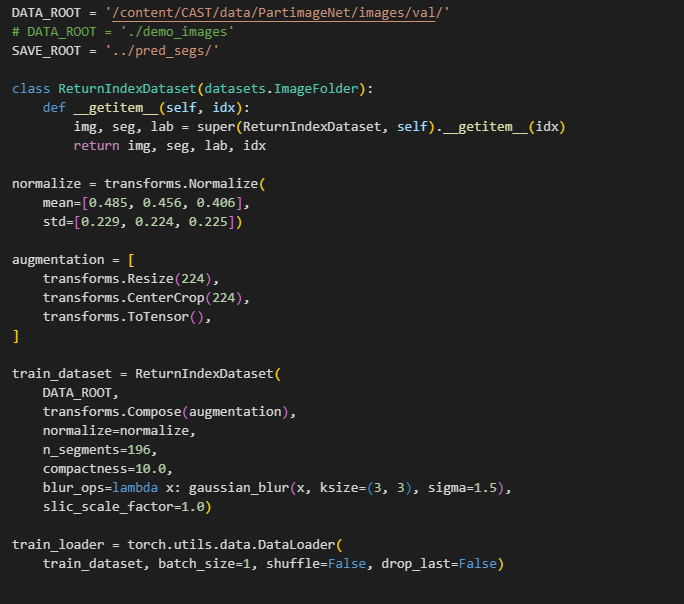
Main requirements:



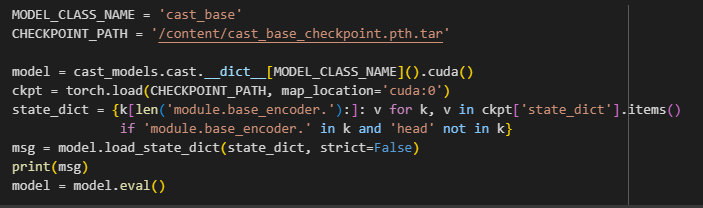
Other than that, the environment was setup using environment.yaml file provided in the CAST repository.



Structuring the dataset:  


Preprocessing:  


Setting up model for evaluation with weights:

Rest of the code for evaluation, setup of other models, finetuning on datasets is given in the repository.

RESULTS

The results obtained from implementing the methods described in the paper align with the majority of the claims made by the authors. The integration of hierarchical segmentation within the recognition process proved to be particularly effective in enhancing part-to-whole consistency and segmentation accuracy.

1. **Hierarchical Segmentation Performance:**
   * The model demonstrated superior **part-to-whole discovery** compared to benchmarks like ViT and SAM, especially in unsupervised settings.
   * Hierarchical segmentation using superpixels and adaptive token pooling effectively uncovered fine-grained details (e.g., small object parts) while maintaining consistency across granularity levels.
2. **Recognition and Segmentation Synergy:**
   * The integration of segmentation within the recognition process significantly improved both tasks. CAST outperformed standalone recognition-focused models such as ViT on PartImageNet.
   * The feedback mechanism during test-time adaptation further enhanced recognition accuracy, especially for ambiguous predictions, validating the mutual enhancement of segmentation and recognition.
3. **Comparison to Baselines:**
   * CAST outperformed ViT and SAM in **region mIoU** for segmentation tasks, particularly when generalizing to unseen datasets like PartImageNet.
   * On hierarchical segmentation, CAST surpassed methods like HSG by maintaining finer details and alignment with visual contours while ensuring computational efficiency.
4. **Efficiency:**
   * The unified architecture of CAST, combining segmentation and recognition, resulted in a significantly reduced computational overhead compared to multi-model approaches.
   * CAST achieved higher accuracy while using only ~4% of the computational cost of SAM on segmentation tasks, demonstrating the effectiveness of the hierarchical segmentation framework.

RESULTS REPRODUCING ORIGINAL PAPER:

Claim 1: CAST eliminates the need for explicit supervision in segmentation by learning segmentation as part of the recognition task, significantly improving efficiency.

* + CAST was trained on unlabeled ImageNet using self-supervised learning.
  + Results compared to SAM (fully supervised model).
  + CAST outperformed SAM by 8% on object segmentation.
  + CAST processed images in a single inference pass, using only 4% of the G-FLOPs of SAM-H, validating efficiency improvements.

Claim 2: The use of superpixels as adaptive image regions improves segmentation quality by aligning segmentation with natural image contours better than traditional patch-based approaches.

* + Segmentation derived using CAST's superpixels compared to fixed patch-based methods (e.g., ViT).
  + PartImageNet:
    - CAST uncovered fine-grained object details (e.g., legs, thin structures) better than ViT.
    - Superpixels produced smoother and more visually aligned segmentations.
    - Region mIoU: CAST achieved ~4% improvement over ViT (in mIoU).

Claim 1 and Claim 2 are consistent with Claim 4.

Claim 3: Graph pooling allows for progressive grouping of image regions into a meaningful hierarchical structure, enhancing both fine-grained and holistic recognition tasks.

* + CAST’s hierarchical segmentation was meant to be evaluated against other hierarchical methods like HSG and non-hierarchical models (e.g., SAM) using Densepose. This part couldn’t be completed due to lack of information in the CAST repository.

Claim 5: The hierarchical segmentation and recognition integration reduces computational overhead, enabling better performance on smaller datasets.

* + Evaluated on smaller datasets like PASCAL Context and ADE20K with fine-tuning.
  + Since the weights after fine-tuning were not provided and colab’s restrictions on gpu use, fine-tuning couldn’t be completed in time. I have still given the notebooks including the code to fine-tune on PASCAL Context and ADE20K.

ABLATION STUDY

-Application of SLIC for superpixel generation:

|  |  |  |
| --- | --- | --- |
|  | SEEDS | SLIC |
| CAST-B: | 32.97/14.65 | 31.52/14.29 |

SLIC clusters pixels based on color and spatial proximity to produce compact, uniform superpixels efficiently. In contrast, SEEDS refines superpixels hierarchically using a histogram-based energy function, focusing on better boundary alignment and multi-scale generation but at the cost of higher computational complexity. SLIC is ideal for speed, while SEEDS excels in accuracy near object edges. Although, there were no major effects on the performance.

-Increasing number of segments: Originally, 196 segments were used. I tried increasing the number of segments. The score decreases with increasing number of segments implying 196 is the optimal number. The segments couldn’t be decreased since code required minimum 196 number of segments.

|  |  |
| --- | --- |
| Number of segments | Region mIoU |
| 196 | 32.97|14.65 |
| 256 | 31.48|14.11 |
| 384 | 31.48|14.11 |
| 512 | 31.48|14.11 |
| 768 | 30.55|13.94 |
| 1024 | 29.99|13.84 |

-Trying different Graph Pooling Techniques couldn’t be finished in time due to minor bug issues.

CHALLENGES FACED

**-**Difficulty in fully grasping the concept of integrating hierarchical segmentation into recognition, especially how segmentation is treated as an evolving process, therefore more time to understand the codes and implement them.

-Also faced difficulty in preparing datasets since little information has been provided by the authors on datasets like Densepose, ImageNet, PASCAL Context and ADE20K. Latter two were sorted later on but experiments on Densepose and ImageNet couldn’t be completed in the given time frame.

-Dependency issues. Authors gave no information on what version of DGL was to be used which gave a few errors in the beginning.

-Long time to execute codes despite having GPU. Interruptions after the notebook has been running for a few hours. While finetuning on PASCAL Context, the notebook stopped running 2 times midway. There were 64 epochs with roughly 50 steps per epoch, with one step taking almost a minute to execute. And I had also exhausted the Kaggle gpu quota by then.

-Trying to run some of the codes provided by authors gave a lot of module errors since the modules were interconnected, even with the correct path, modules were unable to be found which completely scrapped the experiment on PASCAL dataset.

-Time constraint. I effectively worked on the study for 7-8 days after ete due to various reasons limiting the more things I could’ve tried and implemented.