Master's Thesis

Improving Unsupervised Instance detection using dino features

Ajesh Krishnan Kizhakke Menakath

Advisers: Silvio Galesso

Examiner: Prof. Dr. Thomas Brox
Prof. Dr. Frank Hutter

Albert-Ludwigs-University Freiburg
Faculty of Engineering
Department of Computer Science
Chair of Pattern Recognition and Image Processing

August 23th, 2024

Writing period

23. 02. 2023 – 23. 08. 2023

Examiner

Prof. Dr. Thomas Brox

Second Examiner

Prof. Dr. Frank Hutter

Advisers

Silvio Galesso

Declaration

I hereby declare, that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.
I hereby also declare, that my Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

Place, Date	Signature

Abstract

In the evolving realm of deep learning for scene understanding, the traditional dichotomy between grouping and recognition has blurred, thanks to integrated end-to-end training systems. Yet, GroupViT reshapes this landscape by reintroducing explicit grouping in deep networks. This novel bottom-up approach rekindles the essence of semantic segmentation in contrasts with traditional top-down methods. GroupViT, trained on weak supervisory signals from text, showcases impressive results across various datasets, highlighting its efficacy for open-vocabulary segmentation.

Within this thesis, we undertake a dissection of the pretrained GroupViT model, isolating two pivotal components for open-vocabulary segmentation: Visual Grouping and Vision-Text Alignment. Based on our analysis, it becomes evident that there is substantial room for improvement, particularly in Visual Grouping. By leveraging GroupViT's pretrained model and further training on a cleaner, smaller MSCOCO dataset, we observed promising enhancements on the dataset. However, there are trade-offs, compromising performance on downstream dataset such as PASCAL VOC and PASCAL Context.

To address these challenges, this work introduces strategic enhancements. First, we incorporate entropy regularization techniques to improve semantic grouping and visual-text alignment. Second, we propose a non-noisy contrastive loss, countering limitations of training on a smaller dataset and leading to more robust, accurate results.

These systematic refinements furnish a framework to further train the pretrained model of GroupViT on smaller and cleaner datasets for segmentation, while upholding the performance across datasets.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Instance detection is a critical task in computer vision, with applications ranging from autonomous driving to medical imaging. Despite significant advancements, unsupervised instance detection remains a challenging problem due to the lack of annotated data. This thesis aims to improve unsupervised instance detection by analyzing the problems with the current state-of-the-art unsupervised instance detection and segmentation methods.

In the field of Computer Vision, a significant shift has occurred from traditional Convolutional Neural Networks (CNNs) to the transformative capabilities of Transformers, as exemplified by the groundbreaking Vision Transformer (ViT) paper by Dosovitskiy et al. (2020)[1]. Subsequently, the concept of utilizing deep ViT features as dense visual descriptors was introduced by Amir et al. (2021)[2], highlighting the strong semantic information these descriptors provide about instances within an image. Currently, the state-of-the-art method for unsupervised instance segmentation, CutLer, as proposed by Wang et al. (2023)[3], employs DINO features[4] as visual descriptors to effectively identify instances within images.

However, challenges such as the grouping of nearby instances, failure to identify complex background patterns, and the omission of small instances persist even in the current state-of-the-art methods. This thesis aims to investigate these failure cases, mainly based on the CutLer [3] baseline and propose methods to enhance the performance of unsupervised instance detection and segmentation tasks.

1.2 Overview of CutLer

In the field of unsupervised object detection and instance segmentation, the recent work by Xudong Wang et al. introduces Cut-and-LEaRn (CutLER)[3], a novel approach that significantly advances the state-of-the-art. CutLER leverages the capabilities of self-supervised models to identify objects without human supervision, and it enhances this capability to train a high-performance localization model without

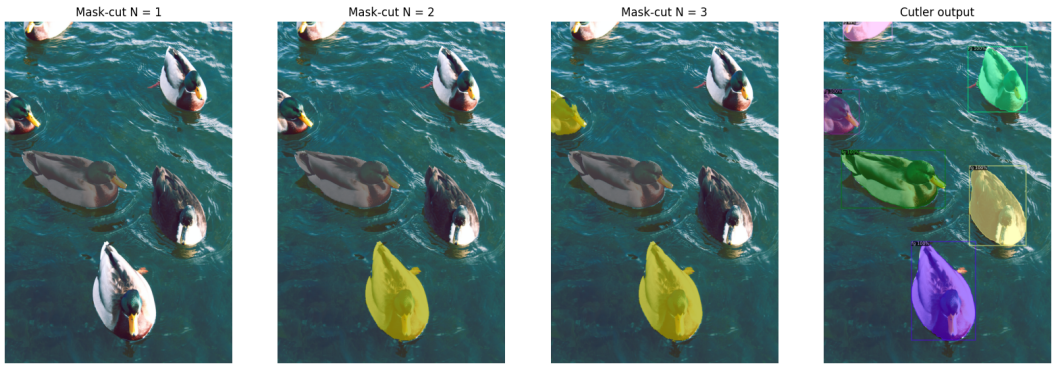


Figure 1: Comparison of maskcut and CutLer outputs. Figure illustrates the outputs of CutLer and maskcut with different N(Number of mask generated) values.

any labeled data. The methodology begins with the MaskCut approach(inspired from [5]), which generates coarse masks for multiple objects within an image. Subsequently, a detector is trained on these masks using a robust loss function. Performance is further improved through a self-training process where the model is iteratively trained on its own predictions. This approach not only simplifies the training process but also proves to be compatible with various detection architectures and capable of detecting multiple objects simultaneously.

The effectiveness of CutLER is demonstrated through extensive evaluations across diverse image domains, including video frames, paintings, sketches, and complex scenes. Notably, CutLER, utilizing a ResNet50 backbone, achieves a substantial performance increase, more than doubling the detection accuracy on 10 out of 11 benchmarks compared to the previous state-of-the-art method, FreeSOLO, which uses a ResNet101 backbone. Specifically, CutLER improves the average precision (AP50) by over 2.7 times across these benchmarks. This demonstrates CutLER's potential not only as a zero-shot unsupervised detector but also as an efficient low-shot detector, marking a significant step forward in unsupervised object detection and instance segmentation.

1.3 Contribution and Key Insights

This study focuses on the shortcomings of CutLer and the methods to minimize them. Our main contributions are:

1. **Analysis on the performance of different metrics for discriminating**

instances: We compare the effectiveness of distance and similarity measures on discriminating instances of same and different classes. +

2. **Analysis on the influence of overlapping instances**: We compare the performance of the model when trained with and without overlapping instances and conclude that training without overlapping instances results in better instance discrimination.

3. **Refining maskcut masks**: On top of self training, ee refine mask-cut masks using CutLer output to remove noisy masks and retraining to yield better performance.

1.4 Outline

- **Chapter 2:**
- **Chapter 3:**
- **Chapter 4:**
- **Chapter 5:**

2 Related Work

2.1 Self-supervised feature learning

Self-supervised feature learning is a crucial process that identifies patterns within extensive unlabeled datasets without the need for human-annotated labels. Plenty of research has been done in this field in the recent years. Several methods have been proposed, each with unique mechanisms and varying levels of success.

2.1.1 Contrastive learning

Contrastive learning has gained significant attention for its effectiveness in self-supervised feature learning. One of the seminal works in this area is SimCLR[6], It employs a simple yet robust framework that leverages data augmentations to create positive pairs from the same image and negative pairs from different images. The model uses a contrastive loss to distinguish between these pairs, learning robust representations in the process. On the other hand, MoCo (Momentum Contrast)[7] introduces a dynamic dictionary with a momentum encoder. This approach allows the model to maintain a queue of negative samples, effectively reducing memory requirements and improving scalability. Nevertheless, it still requires a substantial number of negative samples to function optimally and necessitates careful tuning of the momentum parameter to balance stability and learning efficiency.

2.1.2 Clustering-based feature learning

Clustering-based feature learning approaches automatically uncover the natural groupings of data within the latent representation space. This clustering process helps in understanding the inherent structure of the data by grouping similar data points together based on learned features. Agglomerative Clustering with Self-supervision[8] can capture multi-scale structures and found to be effective for diverse datasets. But found to be computationally expensive and needs careful tuning of the self-supervised task. SwAV[9] combines clustering with contrastive learning by swapping assignments between different augmented views of the image. This method is efficient in terms

of computational resources and achieves state-of-the-art performance on several benchmarks. But it is sensitive to the choice of hyperparameters.

2.1.3 Distillation-based methods

Distillation-based methods have also shown considerable promise in self-supervised learning. BYOL (Bootstrap Your Own Latent)[10] introduces a teacher-student network where the student learns to predict the teacher's representations. Remarkably, BYOL achieves this without using negative samples, simplifying the training process and reducing computational demands. However, it is sensitive to the choice of data augmentations and network architecture, and there is a potential risk of model collapse if not properly tuned. DINO[4], extends the self-distillation approach to Vision Transformers[1]. DINO captures global image representations effectively without relying on negative samples. It shows strong performance on object detection and segmentation tasks, showcasing the potential of transformers in self-supervised learning.

Unlike these unsupervised representation learning efforts, our research revolves around CutLer[3], which focuses on automatically identifying natural pixel groupings and detecting instances within each image.

2.2 Unsupervised object detection and instance segmentation

If we consider the recent methods for unsupervised object detection semantic segmentation, most of them leverage on self-supervised Vision Transformer(ViT)[1] features. DINO[4] observes that the underlying semantic segmentation of images can be extracted using the saliency maps from the ViT.

The quality of this segmentation is superior to the existing methods if the image contains only is one instance. The superiority of DINO features to separate foreground and background has been affirmed by later works[11]. Building on this observation, both LOST [12] and TokenCut [5] utilize DINO features to segment a single salient object from each image. These methods capitalize on the strength of DINO to construct a graph from the features of image patches. Unlike TokenCut and DINO, which can only detect one instance, LOST is capable of finding multiple instances within an image. But it can't be used as a pre-trained model for down stream tasks. But CutLer[3] not only can detect multiple instances, the model can be further used as a pretrained model for label-efficient and fully-supervised learning.

FreeSOLO[13] and the follow up work Exemplar-FreeSOLO[14] (with its addition

of a randomly drawn pool of exemplars used in a contrastive learning loss) generates coarse segmentation masks with low guality and refines it further through self training similar to Cutler. But the poor quality of the coarse maps is a major draw back of this method, where as CutLer masks made by the MaskCut[3, 5] algorithm are usually better in quality and quantity than the initial masks used by MaskDistill[15] and [13]. Even though Maskdistill produces similar quality masks compared to MaskCut, as it only produces one class agnostic mask per image and MaskCut produces N fixed number of masks per image to use as pseudo labels, MaskCut weighs over Maskdistill in quantity.

As CutLer dominates in most cases, including producing better pseudo ground truth masks, ablility to detect multiple instances, compatibility with various detection architectures, usable as pretrained model for supervised detection, our work would mostly focus on studying and improving the performance of CutLer.

2.3 Unsupervised Semantic Segmentation

In the realm of unsupervised semantic segmentation, the challenge of acquiring dense annotations has spurred innovative research efforts. Addressing this constraint, a series of studies [16, 17, 18, 5, 19] have delved into harnessing self-supervised learning techniques to cultivate feature representations capable of supporting segmentation without necessitating extensive annotation efforts. Particularly, the DINO self-supervised model has illuminated that features extracted from its deep layers exhibit noteworthy correlation patterns congruent with genuine semantic labels. This insight has spurred further investigation. Drawing from this, Hamilton et al. introduced STEGO, which employs contrastive loss to distill pre-trained unsupervised visual features into semantic clusters, thus achieving compelling unsupervised segmentation results [19]. Furthermore, Melas-Kyriazi et al. leverage spectral clustering on deep unsupervised representations, achieving state-of-the-art performance while also capitalizing on the informative features derived from DINO's architecture. Further This growing body of work showcases the potential of self-supervised methods, especially those rooted in DINO, to advance the landscape of unsupervised semantic segmentation [17].

3 Background

3.1 Transformers

The Transformer, introduced by Vaswani et al. in 2017 [20], revolutionized sequence-to-sequence tasks in Natural Language Processing(NLP). This architecture overcame limitations of traditional methods like LSTM and gated RNNs [21][22] by harnessing self-attention mechanisms. Self-attention enabled parallel computation, addressing slow training and capturing long-range dependencies. Following the success of Transformer in NLP, Vision Transformer (ViT) was introduced by Dosovitskiy et al. [1] as an alternative to Convolutional Neural Networks (CNNs) for image recognition. This section explores the fundamental components of the Transformer: self-attention, multi-head attention, positional encoding, and feed-forward networks. This section describes the Transformer block utilized within GroupViT, for both textual and visual backbone. A firm understanding of the Transformer is essential to comprehending the functioning of GroupViT.

Input Tokens

In natural language processing (NLP) and vision transformers (ViT), tokens play a crucial role in representing discrete units within an input sequence or image. Tokens can be individual words, subwords, or even characters, depending on the tokenization strategy employed. Tokenization is the process of breaking down a text or image into these discrete tokens. For text data, popular tokenization methods include Byte-Pair Encoding (BPE) and WordPiece[23][24]. BPE and WordPiece algorithms split text into subword tokens, enabling more flexible handling of languages with complex word structures. Each token is then mapped to an embedding vector using an embedding layer. These embedding vectors capture the semantic meaning and contextual information of the tokens within the text. In the case of ViT (Vision Transformers), images are divided into non-overlapping patches, and each patch is treated as a token [1]. These image patch tokens are then converted into embeddings using linear projections. This approach allows ViT models to process and understand

images in a manner similar to how traditional transformer models handle text data.

Positional Encoding

Positional Encoding is crucial for maintaining token order in Transformer models. Often implemented with sinusoidal functions across various frequency dimensions, it enhance the model's grasp of sequential patterns. It is worth noting that ViT employs positional embeddings differently from NLP. In ViT, these embeddings encode the spatial locations of image patches. As a result, they are added to patch embeddings, enabling ViT to capture spatial relationships and contextual information. GroupViT employs simple positional embeddings for its visual and text encoders.

3.1.1 Attention

Attention is a mechanism in the neural network that a model can learn to make predictions by selectively attending to a given set of data. The amount of attention is quantified by learned weights and thus the output is usually formed as a weighted average[25]. For each element, distinct query, key, and value representations are computed. Attention scores are derived from dot products between queries and keys, and they are adjusted by a normalization factor. The application of a softmax function further refines these scores, resulting in attention weights. These weights play a crucial role in shaping the attended representations obtained by applying them to the value vectors

Furthermore, this attention mechanism takes on two key forms: self-attention and cross-attention. Self-attention emerges when the attention source (query) aligns with the attention target (keys and values) within the same sequence. In contrast, cross-attention comes into play when the attention source differs from the attention target.

Assuming learnable weight matrices W_f, W_q, W_k, and W_v in $\mathbb{R}^{D\times D}$:

$$\text{Attention\_Matrix}(q,k) = \text{Softmax}\left(\frac{qW_q(kW_k^T)}{\sqrt{D}}\right) \tag{1}$$

$$\text{Attention}(q,k) = (\text{Attention\_Matrix}(q,k)\cdot vW_v)\,W_f \tag{2}$$

Mathematically, this differentiation can be succinctly described as follows:

$$\text{Attention}(q,k) = \begin{cases} \text{Self-attention}: & \text{Attention}(q,q) \quad (\text{when } q=k) \\ \text{Cross-attention}: & \text{Attention}(q,k) \quad (\text{when } q\neq k) \end{cases} \tag{3}$$

3.1.2 Multi-Head Attention

To enhance expressiveness and capture intricate patterns, Transformers often incorporate the concept of multi-head attention. In this mechanism, several sets of learnable weight matrices for queries, keys, and values are utilized. We define these learnable weights for each head as $W_q^{(i)}$, $W_k^{(i)}$, $W_v^{(i)}$ in $\mathbb{R}^{D \times D_h}$, where D represents the model dimension, i indexes the individual attention head, and D_h represents the dimension focused on by each head. D_h is obtained by dividing D into H chunks equally, where H is the number of heads.

Mathematically, the multi-head attention process can be expressed as follows:

$$\text{MultiHeadAttnMat}(q, k, i) = \text{Softmax}\left(\frac{qW_q^{(i)}(kW_k^{(i)})^T}{\sqrt{D}}\right) \tag{4}$$

$$Attn_h(q, k, i) = \text{MultiHeadAttnMat}(q, k, i)(vW_v^{(i)}) \tag{5}$$

The outputs of the different attention heads are subsequently concatenated and linearly transformed by the projection weight $W_f \in \mathbb{R}^{D \times D}$, resulting in the final output.

$$\text{MSA}(q, k) = [Attn_0(q, k, i), ...Attn_{H-1}(q, k, i)\ ]W_f \tag{6}$$

This strategic combination empowers the model to simultaneously attend to various aspects or relationships within the image. Consequently, a more nuanced representation of the interdependencies between image patches is obtained.

3.1.3 Feed-forward Networks

The Transformer model incorporates feed-forward neural networks to process the output from the self-attention mechanism. These networks consist of two linear layers with a GeLU activation function in between[26]. By incorporating non-linear transformations, the feed-forward networks enable the model to capture complex relationships within the sequence effectively.

$$\text{MLP}(x) = \text{GELU}(xW_1 + b_1)W_2 + b_2 \tag{7}$$

3.1.4 Transformer block

In Chapter 4, GroupViT employs a term called the 'Transformer block' which is an alternative term to the 'Transformer encoder' in the original work by Vaswani et al. [20] and in Dosovitskiy et al.'s Vision Transformer (ViT) [1]. This section delves into

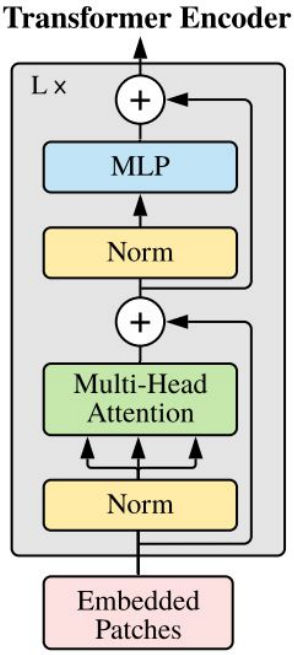


Figure 2: Architecture of Transformer Block. This illustration of the Transformer encoder is present in ViT [1].

a detailed explanation of this fundamental building block.

The Transformer block consists of alternating layers of multi-headed self-attention (MSA) as shown in Eq. 6 and MLP blocks as shown in Eq. 7. Layernorm (LN) [27] is applied before every block, and residual connections [28] after every block as shown from Eq. 8 to Eq. 10, where x represents the input to the Transformer block and y is the final output of the Transformer block after Layer Normalization (LN).

$$\text{Multi-Head Self-Attention (MSA):} \quad x' = \text{MSA}(\text{LN}(x)) + x \tag{8}$$

$$\text{Multi-Layer Perceptron (MLP):} \quad x' = \text{MLP}(\text{LN}(x')) + x' \tag{9}$$

$$\text{Layer Normalization (LN) and Output:} \quad y = \text{LN}(x') \quad \text{(final output)} \tag{10}$$

In the upcoming chapter, we will harness the capabilities of this Transformer block within the GroupViT framework. Our focus will be twofold: firstly, using it to generate image patch embeddings within the visual backbone, and secondly, utilizing it as the text encoder for generating text embeddings within the GroupViT architecture.

3.2 DINO

The self-supervised model DINO, introduced by Caron, Mathilde, et al. [4], demonstrates performance comparable to many state of the art convolutional networks (convnets) trained in supervised manner. Notably, the features extracted from DINO reveal explicit information about the semantic segmentation of images and scene layout. This characteristic stands out in comparison to supervised Vision Transformers (ViTs) and convnets. It's noteworthy that this ability could have valuable applications for weakly supervised image segmentation.

3.2.1 Knowledge distillation

Central to DINO is the concept of knowledge distillation. It involves a student network predicting the teacher network's output. The teacher network is constructed using a momentum encoder, as shown in Figure 3. The optimization objective is to minimize the cross-entropy loss between the probability distributions generated by the teacher ($P_t(x)$) and student ($P_s(x)$) networks, as expressed in Equation 11.

$$\min_{\theta_s} \mathcal{H}(P_t(x), P_s(x)) \tag{11}$$

Here, 'min θ_s' signifies adjusting the student network's parameters θ_s during training. The term $\mathcal{H}(P_t(x), P_s(x))$ represents the cross-entropy loss between the predicted probability distributions. This optimization aligns student and teacher predictions, transferring knowledge effectively.

3.2.2 Multi-crop strategy

DINO generates multiple crops of an image, naturally having diverse perspective, represented as a set $\mathcal{V}$ encompassing various views, including both global views (x_{g1} and x_{g2}). While all crops are processed by the student network, only the global views undergo evaluation by the teacher network. This approach facilitates the establishment of connections that span from local to global features. To go into specific of the Eq. 11 mentioned above, we provide Eq. 12

$$\min_{\theta_s} \sum_{x \in \{x_{g1}, x_{g2}\}} \sum_{x_0 \in \mathcal{V}, x_0 \neq x} H(P_t(x), P_s(x_0)) \tag{12}$$

Here, $H(P_t(x), P_s(x_0))$ denotes the cross-entropy loss between the probability distributions $P_t(x)$ and $P_s(x_0)$, x and x_0 represent input images from the set $\mathcal{V}$, and

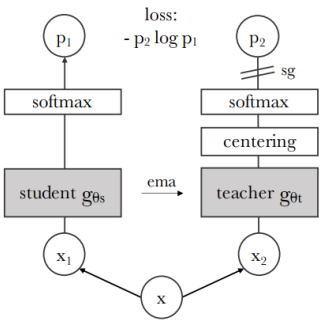


Figure 3: Architecture of DINO. DINO is illustrated in the case of one single pair of views (x1, x2) for simplicity. The model passes two different random transformations of an input image to the student and teacher networks. Both networks have the same architecture but different parameters. The output of the teacher network is centered with a mean computed over the batch. Each networks outputs a K dimensional feature that is normalized with a temperature softmax over the feature dimension. Their similarity is then measured with a cross-entropy loss. We apply a stop-gradient (sg) operator on the teacher to propagate gradients only through the student. The teacher parameters are updated with an exponential moving average (ema) of the student parameters. Illustration provided in [4].

the goal is to minimize the disparity between predictions made by the student and teacher networks.

3.2.3 Momemtum Encoder

To train the teacher network g_{θ_t}, DINO employs an exponential moving average (EMA) on student weights using a momentum encoder [4]. The update rule is given by Equation 13.

$$\theta_t \leftarrow \lambda\theta_t + (1 - \lambda)\theta_s \tag{13}$$

Furthermore, DINO applies centering and sharpening of the momentum teacher outputs to prevent model collapse. Centering prevents one dimension from dominating and encourages uniform distribution, while sharpening has the opposite effect. By balancing these operations, DINO maintains stability while avoiding collapse.
In this work, we explore ways to utilize features extracted from DINO to boost visual grouping in GroupViT.

4 Approach

Conventional deep learning methods typically lack a clear distinction between Grouping and Recognition until the final stages of processing. These models follow a top-down approach, where they classify each pixel into a category, leading to implicit grouping of pixels.

In contrast, Buhmann et al.'s foundational research, which underscores principles from biological vision, highlights the importance of Grouping and Recognition in a more human-like manner [29][30]. Humans tend to first group semantically related concepts and then recognize them. Therefore, the authors emphasize the need for machine vision algorithms that can replicate these functions.

GroupViT, inspired by these biological principles, takes a different approach. It follows a bottom-up strategy, wherein it initially groups image pixels into semantic concepts and then proceeds with labeling through a recognition module. Unlike traditional models that rely solely on pixel-level supervision, GroupViT leverages text to organically group related concepts facilitated by a hierarchical architecture. Importantly, this approach equips GroupViT to handle open-vocabulary segmentation effectively.

Disclaimer: In this chapter, we offer a comprehensive explanation of GroupViT for improved clarity, though it is not a new addition. For simplicity, we refer to Xu, Jiarui, et al. as 'we' here. Our work involves evaluating and improving the baseline model with targeted adjustments. A significant improvement includes enhanced text extraction for multi-label loss, entropy regularization methods, and a noise-free contrastive loss, elaborated from Section 4.7 onwards.

4.1 Comparison of similarity/distance measures in discriminating object class

GroupViT employs multi-stage hierarchy in the vanilla ViT architecture to perform grouping of semantically coherent concepts. Each stage is composed of a Transformer Layer and a Grouping Block. The Transformer layer integrates multiple vanilla

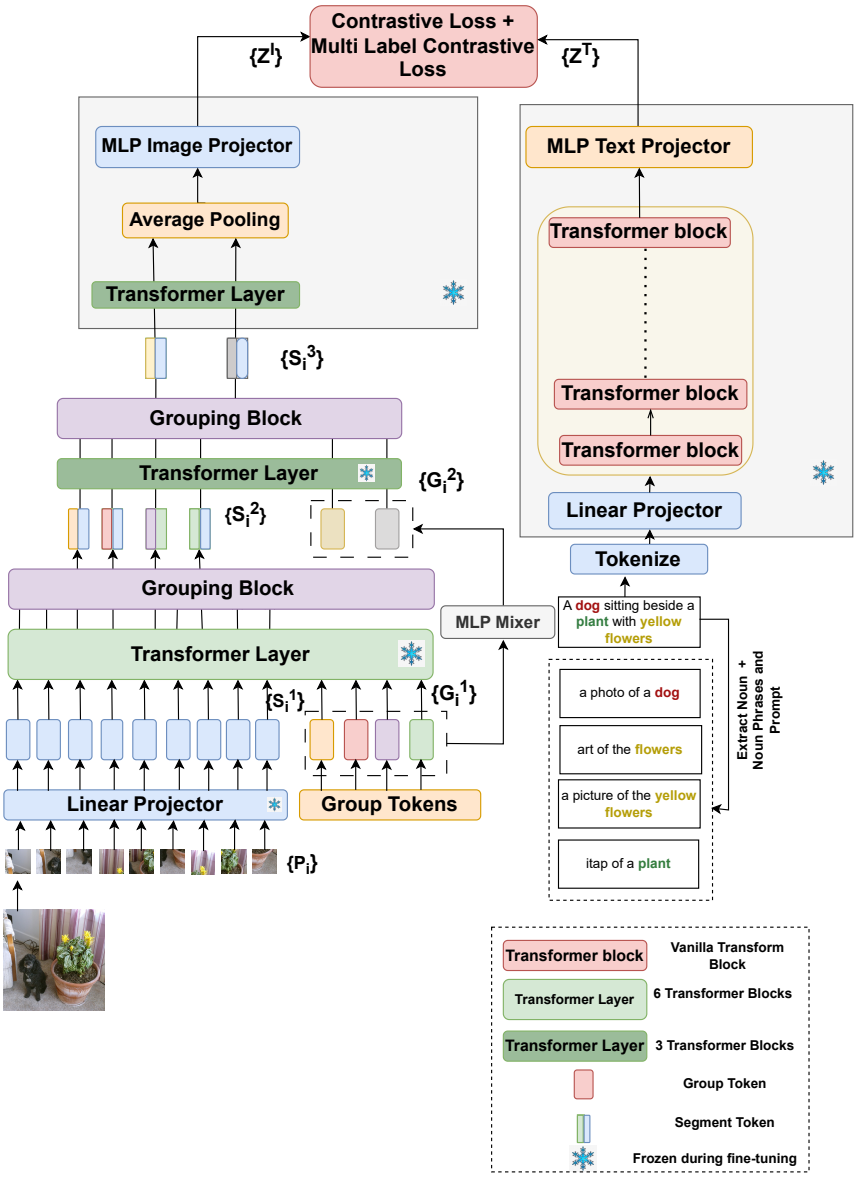


Figure 4: GroupViT: Architecture and Training Pipeline. GroupViT employs a hierarchical structure for grouping. Initial stages introduce learnable group tokens to group coherent image concepts via the Grouping Block. Text Encoder processes caption and nouns extracted from caption. Finally, training employs image-text contrastive and multi-label contrastive loss.

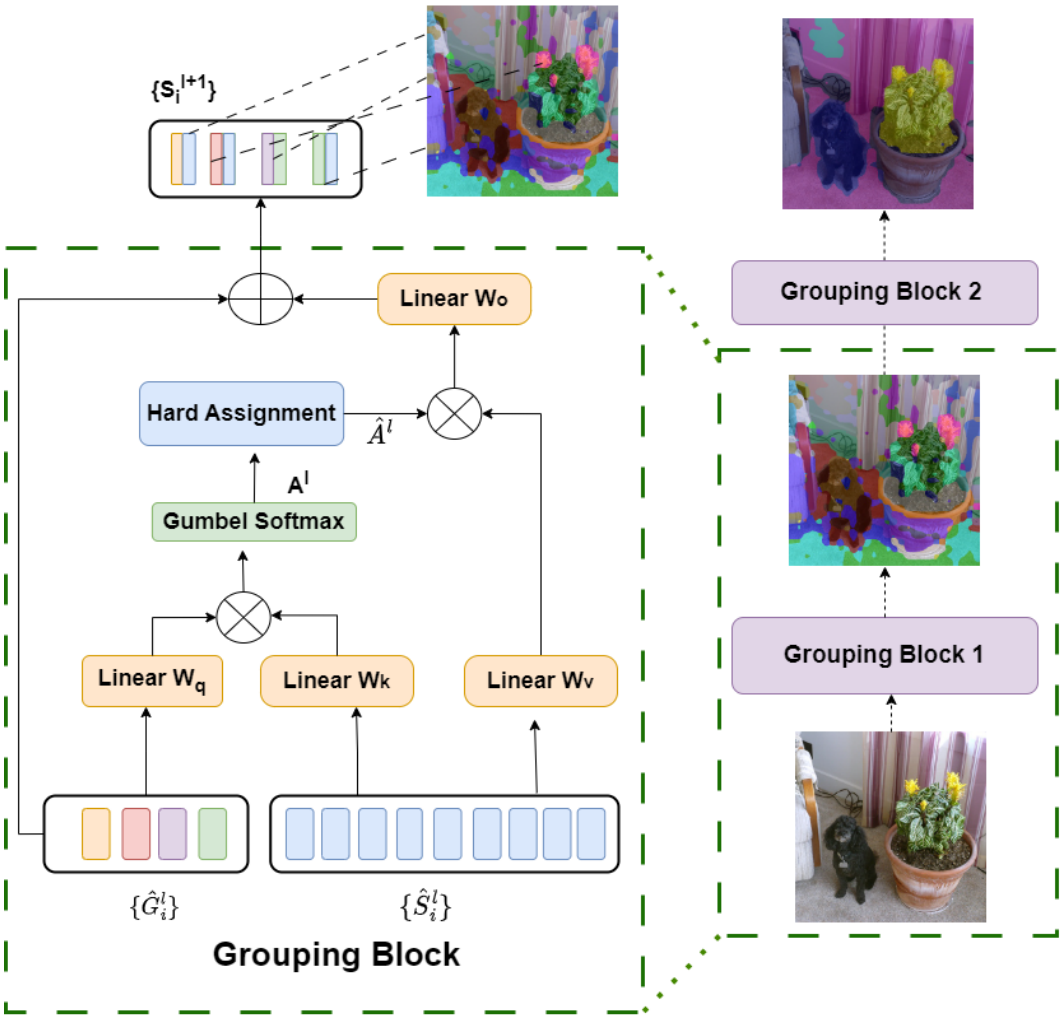
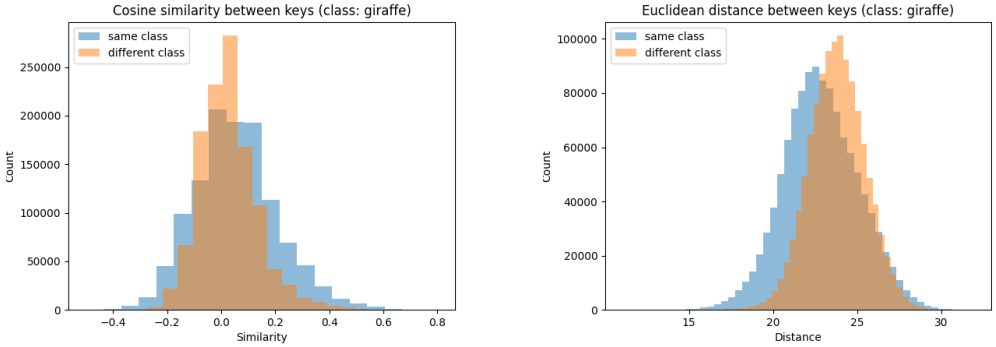


Figure 5: GroupViT: The Architecture of Grouping Block. Each grouping stage ends with a grouping block that computes the similarity between the learned group tokens and segment (image) tokens. The assignment is computed via Gumbel-Softmax over group tokens and converted into a one-hot hard assignment. The segment tokens assigned to the same group are merged together and represent new segment tokens that are input to the next grouping stage.

Transformer blocks [20][1] to execute self-attention that aggregate information globally from all image tokens. Furthermore, a Grouping Block conclude



(a) Cosine similarity **(b)** Euclidean

Figure 6: Comparison of keys of images of same and different classes. Shows the pattern of similarity scores and distance measure when comparing keys of same and different classes. Y axis shows number of images compared

4.2 Relaxed best buddies

GroupViT employs multi-stage hierarchy in the vanilla ViT architecture to perform grouping of semantically coherent concepts. Each stage is composed of a Transformer Layer and a Grouping Block. The Transformer layer integrates multiple vanilla Transformer blocks [20][1] to execute self-attention that aggregate information globally from all image tokens. Furthermore, a Grouping Block conclude´

4.3 Input data

During GroupViT training, we utilize image-text pairs as the training data..

Image

For images, we integrate various augmentation techniques for model robustness and generalization during training. The augmentation process encompasses color jitter, resizing, cropping, and diverse augment transforms to ensure variability and invariance. We also introduce occluded regions through random erasing to enhance occlusion handling. The images are resized to 224×224 pixels, followed by partitioning into non-overlapping 16×16 patches with a stride of 16×16, similar to the Vision Transformer (ViT) methodology [1]. For each image, we split the image into $L = (H \times W)/P^2$ non-overlapping patches with patch size P.

Text

For text, GroupViT extracts nouns from image caption using NLTK [31] and prompts them with a predefined set of templates having 7 prompts, introduced by Radford et al. in CLIP [32]. If NLTK does not extract the required number of labels due to either limited richness in the caption or diversity in the image, GroupViT addresses this by utilizing the associated caption to fulfill the required label count. This ensures a consistent input count for the model.

4.4 Vision Encoder

GroupViT utilizes the ViT backbone, enhanced by additional components designed to facilitate the grouping of semantically coherent concepts at different stages within its architecture. The GroupViT model is structured into three stages, with the first two stages culminating in the incorporation of a Grouping Block dedicated to visual grouping.

In GroupViT's earlier stages, there is a Transformer Layer and a Grouping Block. The Transformer layer integrates multiple vanilla Transformer blocks [20][1] to aggregate global information from all image tokens. For the first and second stages, a Grouping Block follows the Transformer layer, identifying and merging coherent concepts. Unlike ViT, as shown in Figure 4, GroupViT does not process all image patches through each network layer. Instead, it conducts patch grouping and forwards segmented patches to subsequent layers. Further information about visual grouping is presented in upcoming sections.

Input Embedding: Image patches are subjected to a linear projection process, projecting them to a higher-dimensional space of 384 dimensions, denoted by $\mathbf{P} \in \mathbb{R}^{H \times W \times D}$. Here, H and W correspond to 16, and D is 384. We denote this patch set as $\{\mathbf{P}_i\}_{i=1}^{N}$, where N is the number of patch tokens. For positional awareness, positional encoding is integrated into the patch embeddings, utilizing a non-sinusoidal and simple positional encoding scheme. The linear projection employs a single-layer MLP projector with an inner dimension of 384, as shown in Eq. 14.

$$\{\mathbf{S}_i^1\}_{i=1}^{N} = \text{Linear\_Projection}(\{\mathbf{P}_i\}_{i=1}^{N}) \tag{14}$$

The input to each stage is denoted as $\{\mathbf{S}_i^l\}_{i=1}^{M_0}$, where M_0 signifies the segmented tokens grouped by the preceding stage. For the initial stage ($l = 1$), M_0 is equal to

N, as denoted in Eq. 14. We introduce learnable parameters for visual grouping in first and second stage, called as group tokens. These group tokens are represented as $\{\mathbf{G}_i^l\}_{i=1}^{N_0}$, where N_0 denotes the number of group tokens. These group tokens are then linearly projected, as the patch tokens, to generate embeddings $\{\mathbf{G}_i^l\} \in \mathbb{R}^{N_0 \times D}$, as shown in Eq. 15

$$\{\mathbf{G}_i^1\}_{i=1}^{N_0} = \text{Linear\_Projection}(\{\mathbf{G}_i\}_{i=1}^{N_0}) \tag{15}$$

4.4.1 Transformer Layer

The Transformer layer compromises of multiple Vanilla Transformer block, introduced by Vaswani et al. [20]. For stages with group tokens in the visual encoder, we concatenate the segment embeddings $\{\mathbf{S}_i^l\}_{i=1}^{M_0}$ and the group token embeddings $\{\mathbf{G}_i^l\}_{i=1}^{N_0}$, as denoted by the ';' operator in Eq. 16. This concatenated input is propagated through self-attention blocks facilitating information propagation. We denote the output from the transformer layer as $\{\hat{\mathbf{S}}_i^l\}$ and $\{\hat{\mathbf{G}}_i^l\}$ for segment tokens and group token respectively as in Eq. 16

$$\{\hat{\mathbf{G}}_i^l\}, \{\hat{\mathbf{S}}_i^l\} = \text{Transformer}(\{\mathbf{G}_i^l\}; \{\mathbf{S}_i^l\}) \tag{16}$$

4.4.2 Grouping Block

Grouping Block merges all the segment tokens that are assigned to the same group token into a single new image segment, which we call Segment Token. This grouping is based on similarity. For better comprehension, we break down the grouping mechanism into two main steps:(i) computing attention to assess the similarity between individual patch tokens and group tokens, and (ii) assigning each patch token to one of the group tokens based on the derived similarity scores.

Attention

Before Grouping Block, the model de-concatenates the segment token embeddings and group token embeddings. The Grouping Block, depicted in Fig. 5, takes $\{\hat{\mathbf{G}}_i^l\}$ and $\{\hat{\mathbf{S}}_i^l\}$ as inputs to perform cross-attention. For cross-attention, $\{\hat{\mathbf{G}}_i^l\}$ serves as q and $\{\hat{\mathbf{S}}_i^l\}$ serves as k and v. The q, k, and v are subsequently linearly projected using weights $\mathbf{W}_q$, $\mathbf{W}_k$, and $\mathbf{W}_v$ to obtain query, key, and value. Then, we compute the similarity matrix $\mathbf{A}^l$ by Gumbel-Softmax [33][34], as shown in Eq. 17. Here, γ is randomly drawn from the gumbel distribution. The obtained value $\mathbf{A}^l$ is referred as **soft attention**.

$$\mathbf{A}_{i,j}^{l} = \frac{\exp(\mathbf{W}_q \hat{\mathbf{G}}_i^{l} \cdot \mathbf{W}_k \hat{\mathbf{S}}_j^{l} + \gamma_i)}{\sum_{k=1}^{M_l} \exp(\mathbf{W}_q \hat{\mathbf{G}}_i^{l} \cdot \mathbf{W}_k \hat{\mathbf{S}}_j^{l} + \gamma_k)} \quad \text{, where } \gamma \in (0, 1). \tag{17}$$

Assignment

We determine the segment token's assigned group by finding the group with the highest value through the one-hot operation over the soft attention calculated above in Eq. 17. We use the straight-through trick [35] to enable differentiation of the non-differentiable one-hot assignment operation, ensuring smooth gradient flow for training purposes, as demonstrated in Equation 18. Here, sg represents the stop gradient operator.

$$\hat{\mathbf{A}}^{l} = \text{OneHotAssignment}(\text{argmax}(\mathbf{A}^{l})) + \mathbf{A}^{l} - \text{sg}(\mathbf{A}^{l}), \tag{18}$$

The obtained value of $\hat{\mathbf{A}}_l$ is referred as **hard assignment** and reflects the assignment of segment tokens to a single group token.

Once the segment tokens are assigned to various learned groups using the hard assignment, the embeddings of tokens within the same group are aggregated to create a new segment token $\mathbf{S}_i^{l+1}$. For each group, we obtain the weighted sum of segment tokens assigned to that group and finally project the merged features using a learnable linear weight matrix $\mathbf{W}_o$, as shown in Equation 19. Together with $\mathbf{G}_i^l$, we form the output segment tokens for the next stage, denoted as $\mathbf{S}_i^{l+1}$, as shown in Equation 19.

$$\mathbf{S}_i^{l+1} = \mathbf{G}_i^l + \frac{\sum_{j=1}^{M_{l-1}} \hat{\mathbf{A}}_l(i,j) \mathbf{W}_v \hat{\mathbf{S}}_j^{l}}{\sum_{j=1}^{M_{l-1}} \hat{\mathbf{A}}_l(i,j)} \tag{19}$$

We visualize the concepts learned by a few Group Tokens from stage 2 in Fig 7, where we observe Group 0 learns semantics of 'horse' and Group 5 learns semantics of 'person'. It is important to note that one group can learn multiple concepts as visualized in Fig. 8 where we observe Group 0 learns multiple concepts in different images such as 'aeroplane', 'bird', 'horse' and 'sheep'

4.4.3 Multi-stage Information Flow

After detailing the components of each stage, we now present the end-to-end flow of information within the visual encoder. GroupViT follows a three-stage approach: the initial stage consists of 6 transformer blocks and 64 learnable group tokens,

represented as $\{\mathbf{G}_i^1\}$ in Fig. 4, with $\{\mathbf{S}_i^1\}$ segment tokens. The Grouping Block merges all the segment tokens that are assigned to the same group token into a single new image segment, resulting in 64 segments, denoted as $\{\mathbf{S}_i^2\}$ in Fig. 4. The second stage comprises 3 transformer blocks, which take $\{\mathbf{S}_i^2\}$ and 8 group tokens, depicted as $\{\mathbf{G}_i^2\}$ in Fig. 4. The Grouping Block further merges coherent concepts into new segment tokens, generating 8 segments, labeled as $\{\mathbf{S}_i^3\}$ in Fig. 4. In the final stage, 3 transformer blocks process $\{\mathbf{S}_i^3\}$ as input, without any group tokens. This gradual reduction in the number of group tokens aims to create larger and fewer segment tokens through similarity-based grouping. This progression is symbolized by the successive narrowing of the width of Grouping Blocks and Transformer layers from stage 1 to stage 3 in Fig. 4.

On second stage, we also integrate the information of learned group tokens of first stage into fresh learnable tokens of stage 2. To handle the difference in group token numbers in the second and first stage, an MLP-Mixer is applied [36], as shown in Eq. 20, and visualized in Fig. 4

$$\{\mathbf{G}_i^2\} = \{\mathbf{G}_i^2\} + \text{MLPMixer}(\{\mathbf{G}_i^1\}) \tag{20}$$

In the third stage, we do not introduce new group tokens, so we pass the output of the second stage directly to the final stage. Unlike the previous stages, the third stage lacks a grouping block and concludes with the application of Average Pooling on segment embeddings, as shown in Fig. 4. This process results in a condensed image representation which we project into a multi-modal feature embedding space, as visualized in Fig. 4. By doing so, we obtain $\mathbf{Z}^I \in \mathbb{R}^{1\times D_J}$, where D_J signifies the dimensionality of the joint latent space corresponding to a 256-dimensional space, shown in Eq. 21. To accomplish this, a 2-layer MLP projector is employed, having an inner dimension of 4096 and an output dimension of 256.

$$\{\mathbf{Z}^I\} = \text{MLP}(\text{Average\_Pool}(\{\hat{\mathbf{S}}_i^3\})) \tag{21}$$

In conclusion, the multi-stage visual encoder conducts visual grouping, leading to the gradual emergence of fewer and larger coherent concepts. This process culminates in the projection of a singular image embedding obtained through average pooling into the shared joint embedding space.

4.4.4 Text Encoder

The Text Encoder is composed of 12 self-attention Transformer blocks, following the architecture introduced by Vaswani et al. [20]. It employs a dimensional space of 256. The extracted labels and captions, detailed in Section 4.3, undergo tokenization and are subsequently projected into a text dimension space, denoted as D_T, by using a linear projector. We further add positional encodings for sequential information in the generated input embeddings and forward it to the text encoder. This results in embeddings for each label.

Following this, the embeddings proceed through an MLP projector, leading to their projection into a multi-modal feature space. This projection yields $\mathbf{Z}_{K+1}^T \in \mathbb{R}^{\{K+1\}\times D_J}$. Here, $K+1$ denotes 'K' extracted labels and the caption, and D_J denotes the dimensionality of the joint embedding space.

$$\mathbf{Z}_{K+1}^T = \text{MLP}(\text{TextEncoder}(\mathbf{T}_{K+1})) \tag{22}$$

During the training process, the caption embedding serves for contrastive loss, while the embeddings corresponding to the extracted nouns are utilized for multi-label contrastive loss. We consider the end-of-sentence token embedding from the text encoder as the overall representation of the text. We will explain these losses in detail in the next section 4.5.1.

4.5 Training Loss

4.5.1 Contrastive loss

We employ a noisy image-text pair contrastive loss, as previously utilized by Radford et al. for the CLIP model [32] and by Jia et al. for the ALIGN model [37]. In the training process, we consider a batch of size B containing image-text pairs $\{(x_i^I, x_i^T)\}_{i=1}^B$, where x_i^I and x_i^T represent the image and text inputs of the i-th pair, respectively. These inputs are encoded using their respective encoders to obtain embedding vectors Z_i^I and Z_i^T, which are then l2-normalized. The paired image-text data is treated as positive examples, while other pairs serve as negatives. **The image-to-text contrastive loss:**

$$L_{I\to T} = -\frac{1}{B}\sum_{i=1}^{B}\log\frac{\exp(\mathbf{Z}_i^I\cdot\mathbf{Z}_i^T/\tau)}{\sum_{j=1}^{B}\exp(\mathbf{Z}_i^I\cdot\mathbf{Z}_j^T/\tau)} \tag{23}$$

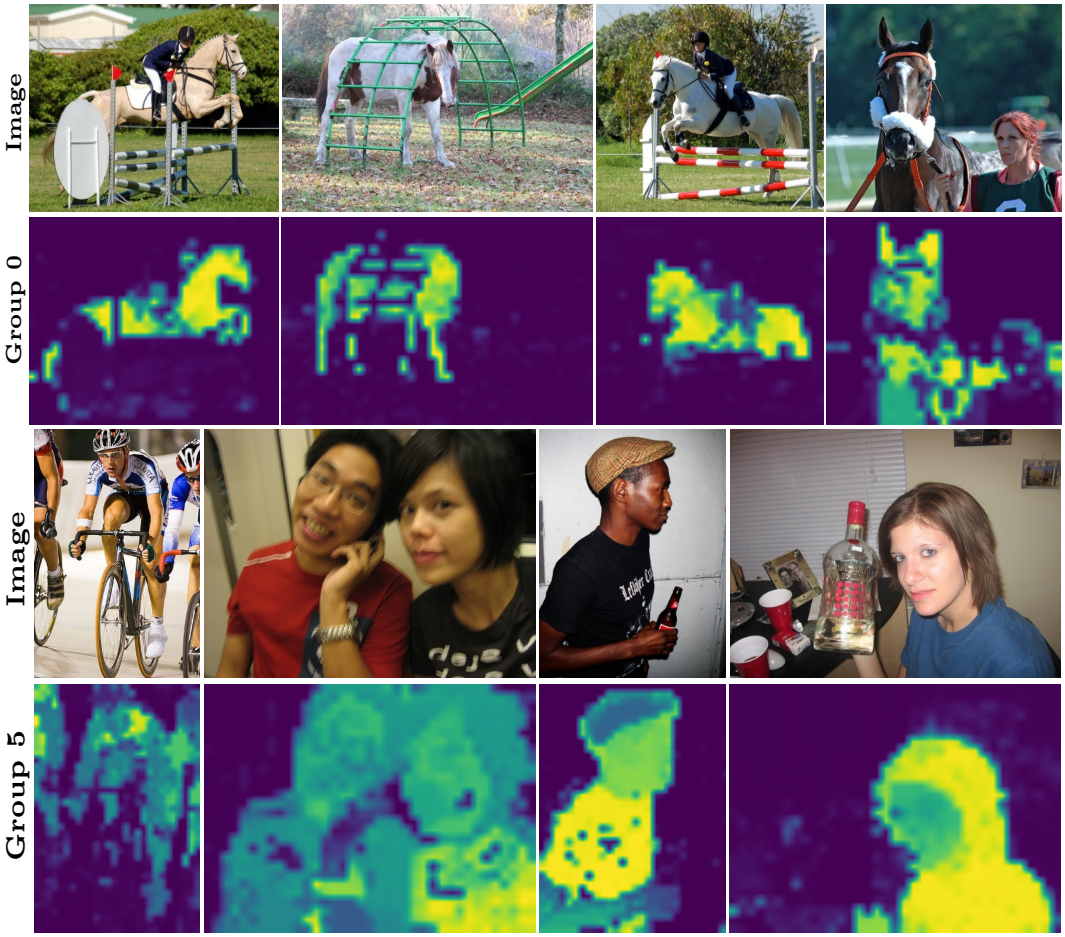


Figure 7: Visualization of concepts learned by Group Tokens in the final stage. We visualize the concepts learned by Group 0 in top row and Group 5 in bottom row. Group 0 learns 'horse', while Group 5 learns 'person'.

Figure 8: Visualization of multiple concepts learned by Group Token 0 in the final stage. We visualize the multiple concepts such as 'aeroplane', 'bird', 'horse' and 'sheep', learned by Group 0, demonstrating that a single group token can capture multiple semantic concepts.

The text-to-image contrastive loss:

$$L_{T \to I} = -\frac{1}{B} \sum_{i=1}^{B} \log \frac{\exp(\mathbf{Z}_i^T \cdot \mathbf{Z}_i^I / \tau)}{\sum_{j=1}^{B} \exp(\mathbf{Z}_i^T \cdot \mathbf{Z}_j^I / \tau)} \tag{24}$$

The bidirectional loss:

$$L_{I \leftrightarrow \{T_k\}_{k=1}^K} = L_{I \to \{T_k\}_{k=1}^K} + L_{\{T_k\}_{k=1}^K \to I} \tag{25}$$

These equations capture the loss functions used in GroupViT for aligning image and text representations during training.

4.5.2 Multi-label Contrastive Loss

In tandem with the Contrastive loss, GroupViT introduces a multi-label contrastive loss to enrich the discriminatory potential of the acquired representation space. This enhancement is particularly relevant for tasks requiring a level of granularity beyond that covered by models like CLIP and ALIGN. To achieve this, GroupViT leverages extracted labels from captions, as discussed in Section 4.3. All the corresponding 'K' prompted labels are employed as positives, while the remaining $B(K-1)$ are treated as negatives within the multi-label contrastive loss framework. While it is plausible that embeddings of identical nouns could be treated as negatives in this scenario, it's essential to emphasize that the ALIGN model underscores the ability of large

web-scaled dataset to offset the presence of noisy image-text pairs. The formulation of this loss is presented in Equation 26.

The image-to-text multi-label contrastive loss:

$$\mathcal{L}_{I\to\{T_k\}_{k=1}^K} = -\frac{1}{B}\sum_{i=1}^{B}\log\left(\frac{\sum_{k=1}^{K}\exp(\mathbf{Z}_i^I\cdot\mathbf{Z}_i^{T_k}/\tau)}{\sum_{k=1}^{K}\sum_{j=1}^{B}\exp(\mathbf{Z}_i^I\cdot\mathbf{Z}_j^{T_k}/\tau)}\right) \tag{26}$$

The text-to-image multi-label contrastive loss:

$$\mathcal{L}_{\{T_k\}_{k=1}^K\to I} = -\frac{1}{KB}\sum_{k=1}^{K}\sum_{i=1}^{B}\log\frac{\exp(\mathbf{Z}_i^{T_k}\cdot\mathbf{Z}_i^I/\tau)}{\sum_{j=1}^{B}\exp(\mathbf{Z}_i^{T_k}\cdot\mathbf{Z}_j^I/\tau)} \tag{27}$$

The bidirectional loss:

$$\mathcal{L}_{I\leftrightarrow\{T_k\}_{k=1}^K} = \mathcal{L}_{I\to\{T_k\}_{k=1}^K} + \mathcal{L}_{\{T_k\}_{k=1}^K\to I} \tag{28}$$

4.6 Inference

The inference process of GroupViT encompasses two primary tasks: (i) identifying visual groups and (ii) assigning labels to these groups. This section provides a comprehensive breakdown of each stage within the inference process.

4.6.1 Class Descriptive Prompts

For inference, we utilize class labels to generate text embeddings. To create class descriptive prompts, we use a simple template: 'A photo of {classname}'. Given N categories, we represent the resulting input embeddings as $\mathbf{T}\in\mathbb{R}^{N\times D_T}$. These embeddings are forwarded to the text encoder and an MLP projector to obtain $\mathbf{Z}^T\in\mathbb{R}^{N\times D_J}$ in the joint feature space, as shown in Eq. 22

4.6.2 Image and Text Alignment

Affinity Metric between Segments and Classes

During inference, similar to the training process, we pass image embeddings through the visual encoder following the approach outlined in Section 4.4. First, as in Eq. 21, we obtain $\mathbf{Z}^I\in\mathbb{R}^{1\times D_J}$, where D_J represents the dimension of the multi-modal feature space. Along with obtaining the average embedding $\mathbf{Z}^I$, we further project $\hat{\mathbf{S}}^3$ into the joint multi-modal feature space, as shown in Eq. 29. The resulting

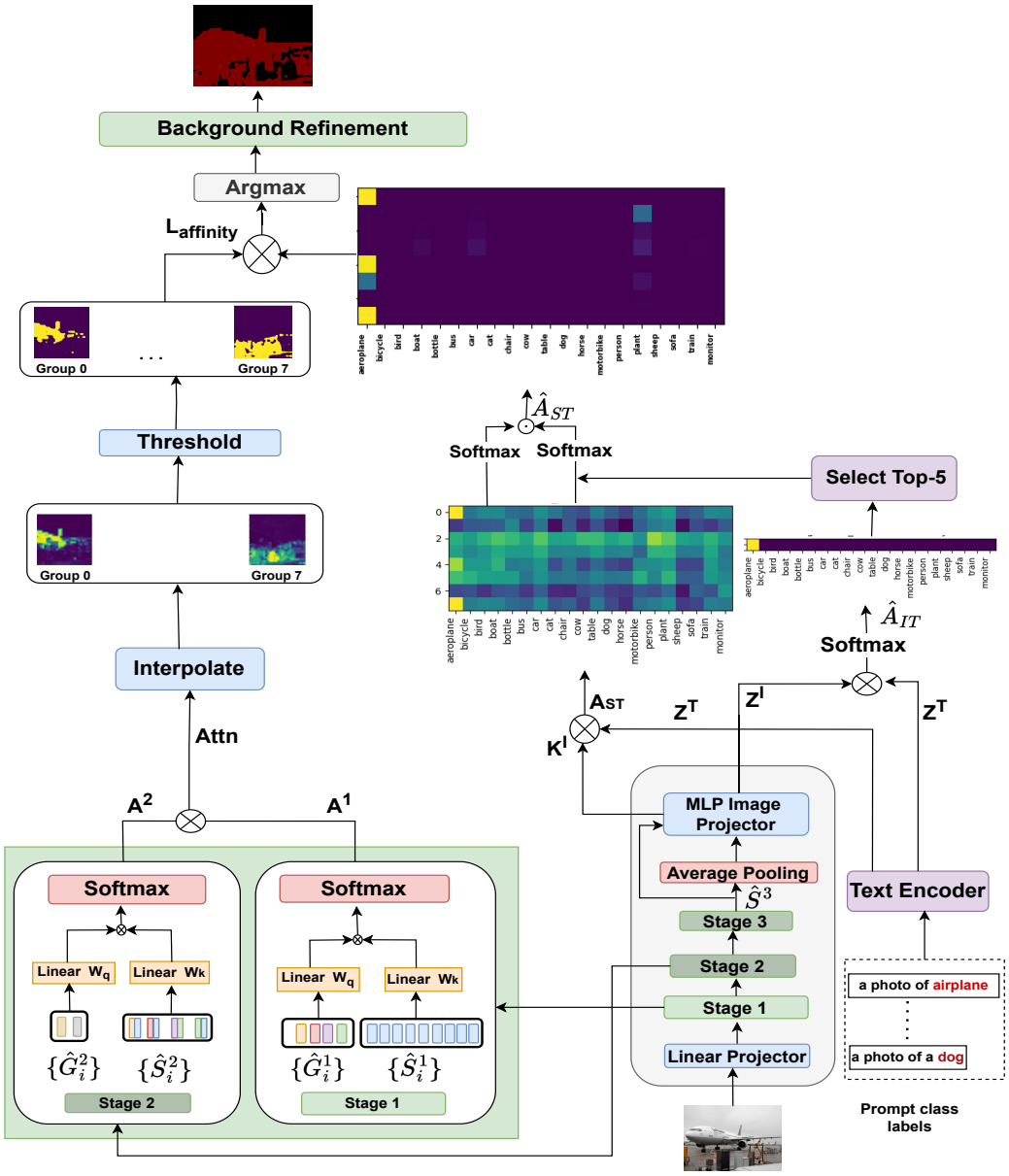


Figure 9: GroupViT Inference Pipeline. GroupViT calculates similarity between the average image embedding and text embeddings ($\mathbf{A}_{IT}$) and between segment embeddings and class label embeddings ($\mathbf{A}_{ST}$). The top-5 classes are selected from $\mathbf{A}_{IT}$, and retained in $\mathbf{A}_{ST}$. After obtaining global attention, it resizes and thresholds the attention maps. Finally, it labels the image based on $L_{affinity}$.

embeddings are denoted as $\mathbf{K}^I$, where $\mathbf{K}^I \in \mathbb{R}^{\hat{S}^3 \times D_J}$, with $\hat{S}^3$ representing the number of segments after the third and final stage.

$$\mathbf{K}^I = \text{MLP}(\hat{\mathbf{S}}_i^3) \tag{29}$$

Following this, we proceed to compute the similarity between the segment embeddings and the class label embeddings in the joint feature space. We label this metric as the '**segment affinity metric**', denoted by $\mathbf{A}_{ST} \in \mathbb{R}^{|S^3| \times N}$.

$$\mathbf{A}_{ST} = \mathbf{K}^I \cdot \text{Transpose}(\mathbf{Z}^T) \tag{30}$$

Average Affinity Metric

We further compute the similarity between $\mathbf{Z}^I$ and the text embeddings $\mathbf{Z}^T$ in the joint feature space. This obtained metric is referred to as the '**average image affinity metric**' and is represented as $A_{IT} \in \mathbb{R}^{1 \times N}$, where N is the number of class labels. This can be seen in Eq. 31

$$\mathbf{A}_{IT} = \{\mathbf{Z}^I\} \cdot \{\mathbf{Z}^T\}^T \tag{31}$$

To convert the obtained values into a probabilistic distribution, we apply the softmax function along the last dimension.

$$\hat{\mathbf{A}}_{IT} = \text{Softmax}(\mathbf{A}_{IT}) \tag{32}$$

Selection of Top-5 Classes

After obtaining the distribution of the average affinity matrix $\hat{\mathbf{A}}_{IT}$, we identify the top 5 classes based on their similarity scores. We retain the similarity values corresponding to only these top 5 classes obtained from $\hat{\mathbf{A}}_{IT}$ in $\mathbf{A}_{ST}$ and set the remaining values in the metric to 0. This process of selecting the top-5 classes and preserving information solely for these classes is referred to as *SelectTop5*, as shown in Eq. 33. We then take a softmax and perform element wise multiplication with the softmax of $\mathbf{A}_{ST}$

$$\hat{\mathbf{A}}_{ST} = \text{Softmax}(\text{SelectTop5}(\mathbf{A}_{ST}, \hat{\mathbf{A}}_{IT})) \odot \text{Softmax}(\mathbf{A}_{ST}) \tag{33}$$

4.6.3 Global Attention

We proceed to obtain the segmented concepts. This involves extracting the soft attention metric from both the grouping blocks, as depicted in Eq. 17. For inference, GroupViT uses Softmax rather than Gumbel Softmax. For simplicity, we keep the same notations.

Moving forward, we obtain soft attention metric for stage 1, denoted as $\mathbf{A}^1 \in \mathbb{R}^{|G^1|\times|S^1|}$, where $|G^1|$ represents the number of group tokens in stage 1, and $|S^1|$ is the number of segment tokens for stage 1. Similarly, we acquire soft attention metric for stage 2, referred to as $\mathbf{A}^2 \in \mathbb{R}^{|G^2|\times|S^2|}$, where $|G^2|$ indicates the number of group tokens in stage 2, and $|S^2|$ corresponds to the number of segment tokens in stage 2. Furthermore, we calculate the similarity between $\mathbf{A}^1$ and $\mathbf{A}^2$. It's important to note that this dot product is feasible because $|G^1|$ is equal to $|S^2|$. As seen in Eq. 34, we obtain the global soft attention over group tokens, denoted as $\mathbf{Attn} \in \mathbb{R}^{|G^2|\times|S^1|}$. This operation can be visualized in Figure 9.

$$\mathbf{Attn} = \mathbf{A}^2 \cdot \mathbf{A}^1 \tag{34}$$

Next, we interpolate these attention maps to inference resolution, obtaining $\mathbf{Attn} \in \mathbb{R}^{|G^2|\times H\times W}$, where H and W are the inference resolution , set to 448. We refine the attention map by assigning each pixel to the most relevant group token G^2 based on the attention values. This process results in masks that indicate the association of each pixel with a single group, as depicted in Figure 9.

$$\hat{\mathbf{Attn}} = \text{Threshold}(\mathbf{Attn}) \tag{35}$$

4.6.4 Integration of Grouping and Labelling

To integrate the visual grouping and visual-text alignment processes, we compute the similarity between the obtained masks $\hat{\mathbf{Attn}} \in \mathbb{R}^{|G^2|\times H\times W}$ and the visual-text similarity metric $\hat{\mathbf{A}}_{ST} \in \mathbb{R}^{|S^3|\times N}$ (Eq. 33), resulting in the matrix $\mathbf{L}_{affinity} \in \mathbb{R}^{H\times W\times N}$.

$$\mathbf{L}_{\text{affinity}} = \hat{\mathbf{Attn}}^T \cdot \hat{\mathbf{A}}_{ST} \tag{36}$$

Subsequently, we apply the argmax operation over the last channel, representing the number of classes, which results in a segmentation map where each pixel is mapped to a specific class.
GroupViT, further introduces a background threshold tailored to each dataset,

taking into account its complexity. If the resulting value after argmax falls below the background threshold, we classify it as the background class and assign it to background. Consequently, if we include the background class, our representation consists of N+1 classes in total to account for each class label. This process is represented as *Background Refinement* in Fig. 9. Now we have the predicted mask and assigned labels with these mask, denoted as **Pred**.This is mathematically represented as follows in Eq. 37

$$\mathbf{Pred} = \text{BackgroundRefinement}(\text{argmax}(\mathbf{L}_{\text{affinity}})) \tag{37}$$

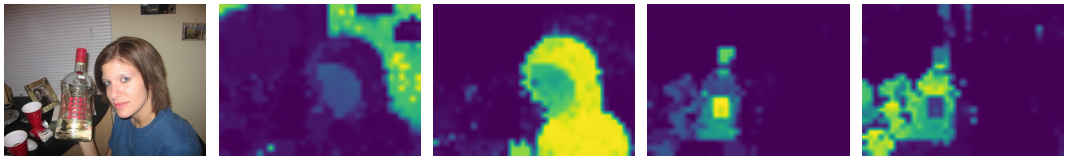
4.7 Entropy Regularization

Now that we have gained a comprehensive understanding of GroupViT, our current objective is to leverage the collective insights accumulated by these groups. To achieve this, we extract the global attention metric $\mathbf{Attn} \in \mathbb{R}^{|G^2| \times |S^1|}$, as outlined in Eq. 34. Subsequently, we proceed with a series of operations, including bicubic interpolation which transforms **Attn** into $\mathbb{R}^{|G^2| \times H \times W}$. In the subsequent stages, we determine the association of each patch by selecting the group with which it has the highest similarity score, achieved through an argmax operation. The culmination of these steps results in a visualization of the emerging groups, as exemplified in Figure **??**.

To understand the region of attention for individual group tokens we now visualize attention maps for each of them in Fig. 10. We observe that multiple pixels of an image often have varying degrees of association to different group tokens. This association of pixels with multiple group tokens raises questions about the confidence of these relationships. For instance, in Fig. 10, 'the face of the woman' is associated with Group 5 as well as Group 3, and pixels representing 'bottle' are associated with Group 0 and Group 7.

4.7.1 Entropy of Patches over Groups

Grounding our motivation on the understanding that, ideally, each patch should align with a single group token, ensuring a clear and unambiguous association, we next calculate the entropy of each patch token over all the group tokens of final stage. For this, we use $\mathbf{Attn} \in \mathbb{R}^{|G^2| \times |S^1|}$, obtained in Eq. 34. Recall, $\mathbf{Attn} \in \mathbb{R}^{|G^2| \times |S^1|}$ represents distribution of segment tokens over group tokens. We visualize computing



(a) Image **(b)** Group 3 **(c)** Group 5 **(d)** Group 0 **(e)** Group 7

Figure 10: Visualization of attention maps of group tokens.

entropy for a single patch token in the Eq. 38 below.

$$H(\mathbf{G}^2, \mathbf{S}_j^1) = -\sum_{i=1}^{|G^2|} \mathbf{Attn}_i \cdot \log(\mathbf{Attn}_i) \tag{38}$$

In this equation, $H(\mathbf{G}^2, \mathbf{S}_j^1,)$ will be maximized when the probabilities produced by the softmax are $1/|G^2|$, indicating that the patch token has an equal probability of association with every group token. It is important to emphasize that our objective is to minimize this entropy. To visually represent the entropy for each pixel and comprehend the entropy of semantically coherent pixels, we interpolate $\mathbf{H} \in \mathbb{R}^{|G^2| \times |S^1|}$ to acquire $\mathbf{H} \in \mathbb{R}^{|G^2| \times H \times W}$. Subsequently, we compute the mean across $|G^2|$ to visualize the entropy maps, as illustrated in Figure 11b.

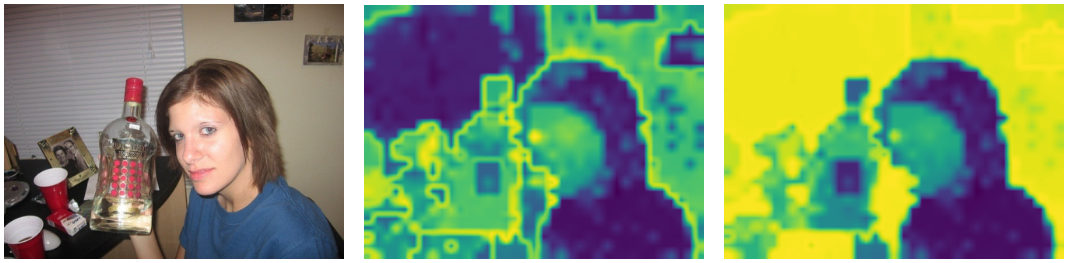
While visualizing the entropy map over group tokens for an image in Fig. 11b, we observe high entropy for objects like 'bottle' and 'face' due to their association with multiple groups. Motivated by this observation, we introduce a penalty term in the loss function, referred to as the Group Entropy Penalty or Group Entropy Loss. This penalty term is calculated as the mean entropy of all patch tokens in an image, as shown in Eq. 39. Subsequently we take the mean penalty for the batch and employ it in conjunction with the contrastive and multi-label contrastive loss.

$$\mathbf{GE} = \frac{1}{|S^1|} \sum_{j=1}^{|S^1|} H(\mathbf{G}^2, \mathbf{S}_j^1) \tag{39}$$

By minimizing this entropy, we aim to enhance the clarity and consistency of the patch token-to-group token associations, ultimately improving the performance of segmenting coherent concepts together.

4.7.2 Entropy of Patches over Labels

Next, motivated by the objective to enhance the alignment between visual and textual cues, we visualize the entropy of image pixels over labels. To do so, we obtain



(a) Image **(b)** Entropy over Groups **(c)** Entropy over Labels

Figure 11: Visualization of Entropy Maps.

$\mathbf{L}_{affinity} \in \mathbb{R}^{H \times W \times N}$, as in Eq. 36. Following this, the entropy of all pixels over labels is calculated as demonstrated in Eq. 40. Here $\mathbf{P}_i$ is the representation of a pixel.

$$H(\mathbf{P}_i, \mathbf{T}) = -\sum_{j=1}^{N} L_{affinity} \cdot \log(L_{affinity}) \tag{40}$$

The resulting visualization of this entropy distribution is presented in Fig. 11c. Given the observation of elevated entropy values in specific object regions, such as a woman's face, we introduce a novel loss function termed the 'Label Entropy Regularization penalty', denoted as LE and defined in Eq. 43. This penalty aims to reduce the entropy of each patch token over the extracted labels during training. During training, we obtain $\mathbf{A}_{ST} \in \mathbb{R}^{|S^3| \times N}$ in Eq. 30, where N would be number of labels extracted. For clarity, in this context, we will use 'K' to denote the number of extracted labels instead of 'N' which was used in Section 4.6 to represent the number of classes. Next we use $\mathbf{Attn} \in \mathbb{R}^{|G^2| \times |S^1|}$, obtained in Eq. 34 to obtain $\mathbf{T}_{affinity} \in \mathbb{R}^{|S^1| \times K}$, as shown in Eq. 41. Next, we calculate entropy of each patch token over labels, as shown in Eq. 43.

$$\mathbf{T}_{\text{affinity}} = \mathbf{Attn}^T \cdot \text{Softmax}(\mathbf{A}_{ST}) \tag{41}$$

$$H(\mathbf{S}_i^1, \mathbf{T}) = -\sum_{j=1}^{K} \mathbf{T}_{affinity} \cdot \log(\mathbf{T}_{affinity}) \tag{42}$$

$$\mathbf{LE} = \frac{1}{|S^1|} \sum_{i=1}^{|S^1|} H(\mathbf{S}_i^1, \mathbf{T}) \tag{43}$$

This penalty aims to refine the alignment of information from both the modalities at earlier stages which could potentially improve performance for a task of finer granularity. We take an average of entropy of all patch tokens of an image and subsequently take a mean of the penalty for the batch. We employ this loss in conjunction with contrastive and multi-label contrastive loss.

4.7.3 Entropy over Segment Affinity Metric

We introduce an entropy penalty for the 'segment affinity metric' computed in the joint embedding space, drawing inspiration from the work of Palepu et al. [38]. First we obtain the metric, denoted as A_{ST}, as shown in Eq. 30.

Next, we apply a row-wise softmax to A_{ST} and compute the entropy between a group token G_i and all of the label embeddings, as illustrated in Eq. 44.

$$\begin{aligned} \text{dist}_{\text{groups}} &= \text{Softmax}(A_{ST}, \text{dim} = -1) \\ H(\mathbf{G}_i^2, \mathbf{T}) &= -\sum_{j}^{K} \text{dist}_{\text{groups}} \cdot \log(\text{dist}_{\text{groups}}) \end{aligned} \tag{44}$$

Here, $H(\mathbf{G}_i^2, \mathbf{T})$ will be maximized when the probabilities produced by the softmax are $1/K$, indicating that the group has an equal probability of association with every label. It's important to note that our objective is to minimize this entropy.

Subsequently, we apply the same procedure to the columns of A_{ST}, we obtain column-wise softmax, followed by entropy calculation for each label embedding over segment embeddings, as depicted in Eq. 45:

$$\begin{aligned} \text{dist}_{\text{labels}} &= \text{Softmax}(A_{ST}, \text{dim} = -2) \\ H(\mathbf{G}^2, \mathbf{T}_j^1) &= -\sum_{i=1}^{|G^2|} \text{dist}_{\text{labels}} \cdot \log(\text{dist}_{\text{labels}}) \end{aligned} \tag{45}$$

These obtained entropy terms are introduced as penalties in our GroupViT method to observe their impact on performance. Additionally, $H(\mathbf{G}_i^2, \mathbf{T})$ is scaled by a factor of 0.2, and $H(\mathbf{G}^2, \mathbf{T}_j^1)$ is scaled by a factor of 0.1 to balance their contributions to the final loss term, denoted by SE, in Eq. 46, following the approach by Palepu et al.

$$\mathbf{SE} = 0.2\left(\frac{1}{|G^2|}\sum_{i=1}^{|G^2|} H(\mathbf{G}_i^2, \mathbf{T})\right) + 0.1\left(\frac{1}{K}\sum_{j=1}^{K} H(\mathbf{G}^2, \mathbf{T}_j^1)\right) \tag{46}$$

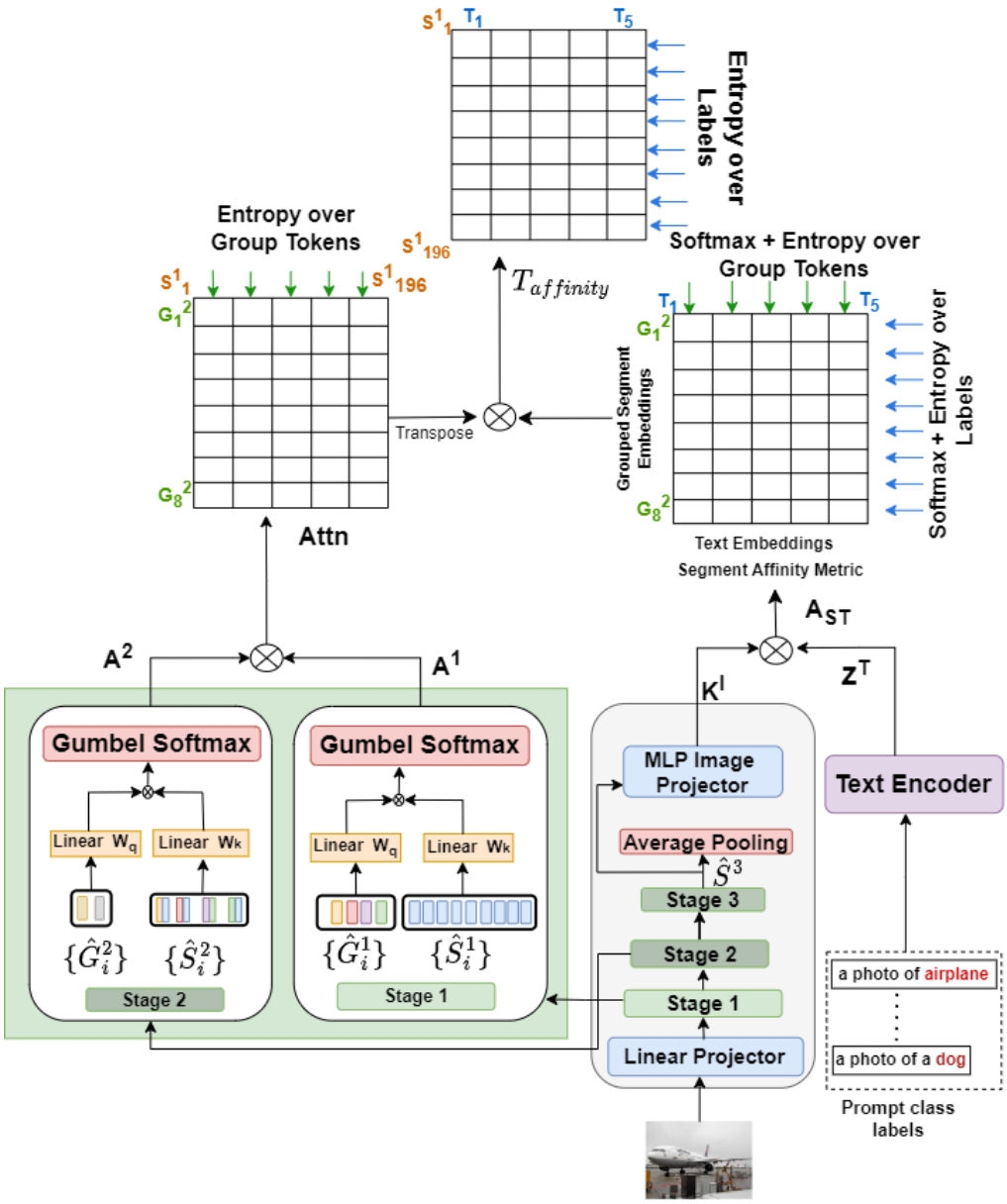


Figure 12: Training Pipeline for Entropy Regularization. We employ global attention to calculate entropy over Group Tokens, which is known as Group Entropy Loss (GE). We use a segment affinity metric to calculate entropy values for both Groups and Labels, collectively referred to as Entropy over Segment Affinity Metric (SE). Additionally, we calculate entropy over labels by using $T_{affinity}$, referred to as Label Entropy Loss (LE).

5 Experiments and Results

5.1 Dataset

[3]

Eval dataset	Imagenet-all	Imagenet-wo-ol	Imagenet-maskcut-filter
Coco Eval	10.35, 19.15	11.52, 20.67	11.44, 21.31
Coco Eval w/o overlapping inst.	23.45, 38.11	24.77, 39.46	24.64, 40.64
Coco Eval only overlapping inst.	7.25, 15.11	8.52, 16.8	8.32, 17.28

Table 1: AP and AP50 of evaluation(box) on COCO Eval datasets on models trained on imagenet for 90K iterations

Eval dataset	Imagenet-all	Imagenet-wo-ol	Imagenet-maskcut-filter
Coco Eval	7.87, 16.05	8.80, 17.47	9.1, 18.15
Coco Eval w/o overlapping inst.	19.19, 35.19	20.13, 36.36	21.15, 38.31
Coco Eval only overlapping inst.	5.12, 11.55	6.18, 13.27	9.78, 18.92

Table 2: AP and AP50 of evaluation(segm) on COCO Eval datasets on models trained on imagenet for 90K iterations

5.1.1 COCO

The COCO dataset, introduced by Lin, Tsung-Yi, et al., has emerged as a prominent and extensively employed resource within the computer vision research community [39]. In our work, we make use of the caption annotations, referred as **MSCOCO**, available from the 2017 splits of the dataset. It encompasses 118,000 training images and 5,000 validation images. Notably, each image in the dataset is accompanied by five captions, resulting in a training set containing a total of 600,000 images. During the evaluation phase, we use 5000 images from the validation split.

5.1.2 PASCAL VOC

PASCAL VOC 2012, introduced by Everingham, Mark, et al., comprises 20 object classes along with a background class, featuring a total of 1,500 training images and 1,500 validation images [40]. In our analysis of GroupViT's features, we use the training split of this dataset, as explained in section 5.4. Additionally, we employ the validation split for zero-shot segmentation in all our experiments.

5.1.3 PASCAL Context

The PASCAL Context dataset is another significant resource, introduced by Mottaghi, Roozbeh, et al., widely utilized in the field of computer vision research[41]. The dataset provides per-pixel segmentation annotations for both object and stuff categories, encompassing 5,000 training images and 5,000 validation images. The images represent a diverse collection of indoor and outdoor scenes, captured from various viewpoints within the PASCAL VOC dataset (PVOC). Unlike the PASCAL VOC dataset that annotate only object classes (for e.g. 'airplane', 'dog', etc.), PASCAL Context annotates contextually relevant stuff categories, along with object classes, enriching scene understanding (for e.g. 'grass', 'ground', 'water', etc.). For our zero-shot evaluation, we use the version with the most frequent 59 classes called as PC-59 which is widely used in the existing literature.

5.1.4 ADE 20K

ADE20K, introduced by Zhou et al., provides detailed annotations for pixel-level segmentation, covering both objects and stuff categories [42]. The dataset is split into a training set and a validation set. The training set consists of approximately 20,000 images, while the validation set contains around 2,000 images. ADE20K is known for its comprehensive coverage of scenes and contains annotations for 150 object classes and 92 stuff categories, making it a valuable resource for zero-shot semantic segmentation in our experiments.

5.2 Evaluation Metric

5.2.1 Mean Intersection Over Union

The Mean Intersection over Union (mIoU) is a widely employed evaluation metric within the field of computer vision, specifically designed for the assessment of semantic segmentation models. It quantifies the degree to which the predicted segmentation

masks align with the ground truth annotations for a given semantic segmentation task involving N classes. This alignment is determined by comparing the prediction logits and ground truth logits for a specific class, denoted as 'c', based on three key parameters: (i) True Positive (TP(c)): Number of pixels correctly classified as class c, (ii)False Positive (FP(c)): Number of pixels incorrectly classified as class c but should belong to other classes, (iii) False Negative (FN(c)): Number of pixels incorrectly classified as other classes but should belong to class c. The IoU for a specific class 'c' is calculated as shown in the below Eq. 47

$$\text{IoU}(c) = \frac{\text{TP}(c) + \text{FP}(c) + \text{FN}(c)}{\text{TP}(c)}, \tag{47}$$

The final mIoU is then obtained by calculating the mean across all classes as shown in the below Eq. 48 to calculate the final mIoU.

$$mIoU = \frac{1}{N} \sum_{c=1}^{N} IoU(c), \tag{48}$$

This metric allows researchers to effectively evaluate the performance of their semantic segmentation models and assess how well the model aligns with the ground truth

5.3 Experiment Setting

The GroupViT model is trained on a combination of CC12M and filtered YFCC datasets, amounting to a total of 26 million image-text pairs [43, 44]. This training spans 30 epochs and employs 16 NVIDIA V100 GPUs over a 2-day period, utilizing a batch size of 4096. The initial two epochs serve as warm-up stages, gradually increasing the learning rate from an initial value of 4×10^{-6} to reach a base of 1.6×10^{-3}. The training process integrates a weight decay of 0.05 to counter overfitting, and gradient clipping to prevent gradient magnitudes exceeding 5.0. Employing the AdamW optimizer with epsilon set at 1×10^{-8} and beta coefficients of 0.9 and 0.999, a cosine learning rate scheduler is employed [45, 46].
For our experiments, we always train the pretrained model on MSCOCO. *In this context, we use 'training' and 'fine-tuning' interchangeably.* With a global batch size of 256, we introduce a learning rate scale of 0.01 to adjust minimum and warm-up learning rates, aiding convergence. The warm-up epoch of 1 allows gradual increase of initial learning rate. A cosine scheduler over 12 decay epochs is used, gradually reducing the learning rate with 2 cycles of annealing for refinement. If not explicitly

mentioned, we use 2 NVIDIA 1080Ti GPUs for our training procedure. We use seed '123' for reproducibility if it is not explicitly mentioned.
For inference, we keep an image resolution of 448, stride of 224 and crop size of 448. For background threshold, we use 0.95 for PASCAL VOC, 0.9 for COCO, 0.35 for PASCAL Context and 0.95 for ADE20K. Note that authors of GroupViT do not evaluate on ADE20K. Therefore, we keep all settings unchanged and use the background threshold as determined by the authors of OVSegmentator [47]. We do an ablation on these inference settings for our baseline to report the best possible setting in section 5.11

5.4 Visual Grouping vs Visual-Text Alignment

In our quest to enhance the performance of the GroupViT approach and identify its limitations, we conduct a comprehensive analysis using the PASCAL VOC dataset. Within GroupViT, the Semantic Segmentation task consists of two key subtasks: visual grouping and visual-language alignment. To precisely identify GroupViT's limitations, we perform two distinct sets of experiments. In each set, we isolate and evaluate these subtasks individually.

5.4.1 Analysis of Visual Grouping

In the first set of experiments, we evaluate GroupViT's performance exclusively on the Visual Grouping task. We refrain from using GroupViT's labeling mechanism and instead propose an alternative approach, as explained below.

Feature Extraction

We extract features using the pretrained GroupViT model for the training split of the Pascal VOC 2012 dataset. Images are processed through the visual encoder with frozen weights, where they are grouped into 8 segments in the final stage. We use features from the final stage, denoted as $\hat{S}_i^3$, to build a feature bank represented as $\mathbb{F} \in \mathbb{R}^{|\hat{S}^3| \times D}$. In our setup, D represents the visual encoder's dimension, which is 384, and $|\hat{S}^3|$ represents the number of segment tokens obtained after the final stage, which is 8. This configuration results in 8 features per image. For our experiment, the feature bank contains a total of 11,592 features.

Soft Label Assignment

To label the features, we deviate from GroupViT's label assignment method. Similar to the inference process outlined in Section 4.6, we compute the global soft attention of group tokens, denoted as $\mathbf{Attn} \in \mathbb{R}^{|G^2| \times |S^1|}$, as shown in Eq. 34. The attention map is then reshaped into a $|G^2| \times P \times P$ format, where $P \times P$ corresponds to a 16×16 patch token format. We use bilinear interpolation to resize the map to $G^2 \times H \times W$, where H and W represent the image's height and width. By performing an argmax operation, we assign pixels to specific groups, resulting in a 2D vector of size $H \times W$ that indicates group associations, as demonstrated in Eq. 49. This process yields a segmented image.

$$\text{Segmented\_Image} = \text{Argmax}(\text{Interpolate}(\mathbf{Attn})) \tag{49}$$

To label the segments in the obtained segmented image, we extract their corresponding labels from the ground truth mask, denoted as M. To address cases where some pixels of a feature are labeled as class 'A' while others are labeled as class 'B,' we implement a soft label assignment approach. This method captures varying degrees of association, offering a more flexible representation of visual content.

To achieve this, we maintain a soft label vector $S \in \mathbb{R}^{|\hat{S}^3| \times (N+1)}$, where $|\hat{S}^3|$ is the number of segments, and $N+1$ represents the class count, including the background class. For instance, in the case of Pascal VOC 2012 with 21 classes, each image feature is associated with a 21-sized array, where the indices correspond to class assignments. If all the pixels of a feature belong to class 'A', the corresponding index is set to 1. When out of N pixels, X pixels belong to class 'A' and (N-X) pixels belong to class 'B', the indices for class 'A' and class 'B' would have values X/N and (N-X)/N, respectively. Each index ideally represents the percentage of segment area associated with a specific class.

Find K-Nearest Neighbors (KNN)

After obtaining soft labels for the features, our goal is to train a K-Nearest Neighbors (KNN) model [48]. To ensure balanced features, we set a maximum limit of 1000 features for each category. We also conducted ablation experiments on the value of K, as shown in Table 4, and present the results in Table 3.

Evaluation

To evaluate the validation set, we follow the same procedure as for the training set to extract group token features. Let V be the set of images in the validation set. For each image $v \in V$, we extract features of its segment tokens $F_v = f_{v1}, f_{v2}, \ldots, f_{v8}$, where f_{vi} represents the ith segment token feature of image v.

Next, we find the nearest neighbors for each feature in the validation set. We use T to denote the bag of features obtained from the training set. For each segment token feature f_{vi} of an image v, we find its k nearest neighbors $KNN_{vi} = t_{vi1}, t_{vi2}, \ldots, t_{vik}$ from T. We aggregate their corresponding soft labels S_{vik} using the mean operation:

$$\overline{S_{vi}} = \frac{1}{k} \sum_{t_{vik} \in KNNvi} S_{vik} \tag{50}$$

We then apply the argmax operation to the aggregated soft labels $\overline{S_{vi}}$ to obtain the final label y_{vi} for each segment token feature f_{vi}:

$$y_{vi} = \arg\max(\overline{S_{vi}}) \tag{51}$$

This process provides us with the final labels for all segment token features in the validation set. Using these labels, we obtain the segmentation logits and evaluate the segmentation results using the mean Intersection over Union (mIoU) metric. The results are presented in Table 3.

In summary, the procedure entails the extraction of segment token features from the final stage for the training split of the PASCAL VOC dataset to create a bag of features. We fit a KNN model on these features. We then extract the features for the validation split. Nearest neighbors are identified for all the features, and soft labels are aggregated, allowing for the assignment of final labels. We then obtain segmentation logits, and evaluate the mIoU score on the dataset. This holistic process offers valuable insights into the model's performance in terms of visual grouping.

5.4.2 Analysis of Vision-Text Alignment

We analyze the performance of the Vision-Text alignment mechanism of GroupViT on PASCAL VOC without utilizing visual grouping, relying solely on ground truth masks. Following the inference pipeline in Section 4.6, we compute the metric $\mathbf{L}_{affinity} \in \mathbb{R}^{H \times W \times N}$ as defined in Eq. 36, where H and W represent the image's height and width, respectively, and N is the number of classes.

Next, we load the ground truth mask for an image, where each pixel is labeled

with its associated class, including the background. Let M be the ground truth mask, with each entry $M_{i,j}$ representing the class label at pixel (i, j). For each class c in M, we gather all pixels belonging to class c in $\mathbf{L}_{affinity}$ to identify groups in $\mathbf{L}_{affinity}$ without employing GroupViT 's grouping mechanism. Specifically, all pixels associated with class 'c' form a group, denoted as G_c. We calculate the mean of this group of pixels, M_{G_c}, along the last dimension (representing classes) to obtain a mean channel value for all pixels in the group. We then perform an argmax operation over the class channel to obtain C_{G_c}, as shown in Eq. 53. Here, C_{G_c} refers to the class assigned to all pixels in group G_c:

$$M_{G_c} = \frac{1}{|G_c|} \sum_{p=1}^{|G_c|} \mathbf{L}_{affinity}(p) \tag{52}$$

$$C_{G_c} = \text{Argmax}(M_{G_c}) \tag{53}$$

We then label all associated pixels with the obtained class C_{G_c}. This way, we identify groups by grouping all pixels belonging to the same class based on the ground truth mask, without using the grouping mechanism of GroupViT . We assign class labels using the GroupViT mechanism, and the results are presented in Table 3. We note that while Visual Grouping and Visual-Text alignment both outperform GroupViT individually, it is evident that Visual Grouping exhibits a larger potential for improvement compared to alignment. We also conducted an evaluation without the background class to report the mIoU score, eliminating the reliance on the susceptibility of the background threshold.

Model	Type	PASCAL VOC
GroupViT	Original	52.29
GroupViT	Visual Grouping	54.16
GroupViT	Visual Grouping(WB)	56.12
GroupViT	Visual-Text Alignment	**76.18**

Table 3: Analysis of Individual Components. Here, 'Visual Grouping (WB)' refers to the evaluation performed without considering the 'background' class.

Figure 13: Visualization of Visual Grouping Vs Visual-Text Alignment. Comparison of Segmentation Masks using Different Models. The first and last columns show image and the corresponding ground truth segmentation masks, while the remaining columns display the segmentation masks generated by Visual-Text alignment, Visual grouping and the Original checkpoint.

Model	K	PASCAL VOC
Visual Grouping	5	51.85
Visual Grouping	10	54.26
Visual Grouping	15	51.93
Visual Grouping	25	49.66

Table 4: Ablation on number of nearest neighbors for Visual Grouping

5.5 Fine-tuning the Pretrained Model

In the previous section, our feature analysis highlighted opportunities to enhance GroupViT's visual grouping capabilities. These insights have motivated us to explore its full potential further. However, GroupViT is a substantial model with 55 million parameters, trained on a large dataset. Starting training from scratch would be resource-intensive. Instead, we've chosen to fine-tune the pretrained model, leveraging the valuable knowledge it acquired during its initial training phase and its understanding of visual and linguistic concepts. This approach also helps accelerate convergence. Our fine-tuning process involves training on a smaller and cleaner dataset, MSCOCO. This dataset provides human-annotated captions, which are believed to offer richer information about the corresponding images compared to the web-scaled datasets used in pretraining.

5.5.1 Choice of Components

To comprehensively analyze the core components of GroupViT for segmentation, we propose fine-tuning the Grouping Blocks and MLP Projectors. This aims to improve visual grouping quality and alignment in an image-text pair. We systematically evaluate different combinations within the visual encoder and report results in Table 5. The experiment was conducted over 15 epochs. We observe that tuning both Grouping Blocks results in better performance compared to the other combinations. This validates our earlier findings in Section 5.4, where we identified a greater room for improvement in visual grouping.

5.5.2 Choice of Batch Size

In our experiments, we change the global batch size to three different values: 256, 512, and 1024 and train the model for 10 epochs. In Table 6, we observe improved performance with a batch size of 1024 on PASCAL VOC and PASCAL Context, while a batch size of 512 works best on COCO, as detailed in Table 6. While the literature suggests that a larger pool of negative samples can enhance contrastive loss outcomes [37][32], it's also noted that large data scales are needed to counteract the impact of noisy image-text training [37]. Our hypothesis aligns with our specific context: training on a relatively smaller yet cleaner dataset. This suggests that the noise introduced during training is not sufficiently mitigated, making a relatively smaller batch size of 512 a more effective choice.

Due to resource constraints and only marginal differences in model performance

Model	Fine-tuned components	Datasets			
		COCO	PVOC	PContext	ADE20K
GroupViT	Original	24.3	**52.29**	22.39	**8.59**
GroupViT	Both GBs	**26.15**	49.18	21.87	7.50
GroupViT	2nd GB	25.79	49.15	21.71	7.69
GroupViT	Both GBs + Img Proj	25.62	49.22	**22.41**	8.49
GroupViT	2nd GB + Img Proj	25.79	50.90	22.33	8.25
GroupViT	Both GBs + Projs	25.05	48.33	22.39	7.61
GroupViT	2nd GB + Projs	25.04	47.94	21.7	7.67
GroupViT	Img Proj	24.48	50.32	22.36	8.27
GroupViT	Projs	24.49	50.24	21.87	8.15

Table 5: Choice of Architectural Components to Fine-tune. Here, 'GBs' stands for Grouping Blocks, while 'Img Proj' represents the Image Projector within the Visual Encoder. 'Projs' refers to both the Text MLP Projector and the Image MLP Projector. '+' is used to represent a combination of components.

Model	Batch Size	Datasets			
		COCO	PVOC	PContext	ADE20K
GroupViT	256	25.96	48.67	21.64	7.60
GroupViT	512	**26.13**	49.29	21.63	**7.92**
GroupViT	1024	25.92	**49.73**	**21.71**	7.57

Table 6: Choice of Batch Size

across different batch sizes, we choose to proceed with a batch size of 256 for our experiments. However, as we observe some improvement with a batch size of 512, we will report the performance of our baseline, obtained later in Section 5.10, using a global batch size of 512.

5.6 Exploring Impact of Multi-label

We investigate how extracted nouns affect performance, particularly their impact on embedding context. To enhance context, we extract comprehensive noun phrases from captions using NLTK and a regex expression. These phrases include adjectives before nouns and are denoted as 'NP' in Table 7. Furthermore, we experiment with varying the number of extracted nouns and phrases (3, 5, 7 labels) to assess their effect on performance.

In addition, we train a model without multi-label contrastive loss, and the results are shown in Table 7. Notably, we observe that the model trained on Noun Phrases, Nouns, and captions, with an upper limit of 5 on extracted labels, outperforms other settings. However, intriguingly, we observe similar performance for the model trained solely with contrastive loss and the one trained with both multi-label contrastive loss and contrastive loss.

To provide a more robust analysis, we conduct experiments on 3 different seeds (0, 100, 200) for 'NP+Nouns+Captions with 5 labels' and 'Caption'. The results are summarized as mean and variance mIoU in Table 7. Visualizations in Figure 25d confirmed convergence to similar performance, with slight initial speed-ups for multi-label training.

As a result, we decide to proceed with the 'NP+Nouns+Caption' model with 5 labels for subsequent experiments. This choice aligns with Xu et al.'s findings during GroupViT training [49].

5.7 Refinement in Label Extraction

For multi-label loss, GroupViT extracts a defined number of nouns or noun phrases from the caption. However, when the number of extracted labels fall short of this defined count, we address this by duplicating the caption. This ensures that the required number of labels is reached, maintaining consistent input lengths across all instances. To enhance clarity and reduce noise, we propose to substitute the caption with the <PAD> token. As <PAD> tokens are primarily introduced to ensure

Model	#labels	Type	Datasets			
			COCO	PVOC	PContext	ADE20K
GroupViT (Original)	3	Nouns+caption	24.3	**52.29**	**22.39**	**8.56**
GroupViT	3	Nouns +Caption	26.32	49.65	22.04	7.95
GroupViT	3	NP+Nouns +Caption	26.35	50.24	**22.09**	**8.17**
GroupViT	5	NP+Nouns + Caption	**26.74** ± **0.0462**	49.80 ± 0.0042	21.92 ± 0.0061	7.42 ± 0.0022
GroupViT	7	NP+Nouns +Caption	25.67	48.11	21.78	7.66
GroupViT	0	Caption	26.61 ± 0.0175	51.79 ± 0.0769	21.83 ± 0.0049	7.45 ± 0.0104

Table 7: Exploring the Impact of Text Hierarchy. The model 'NP+Nouns+Caption' extracts noun phrases and nouns for use in multi-label contrastive loss and captions for contrastive loss, while the model 'Nouns+Caption' extracts only nouns for use in multi-label contrastive loss. The model 'Caption' is trained without multi-label contrastive loss.

uniform input length, their embeddings are devoid of meaningful content. Therefore, we suggest assigning null values to their embeddings, encouraging the model to focus exclusively on pertinent input information and minimizing bias introduced by padding tokens. Our experiments, spanning over 30 epochs, reveals an improvement in performance, as indicated in Table 9. This discovery motivates us to adopt the practice of embedding <PAD> tokens with void embeddings. We provide a qualitative analysis of this baseline on PASCAL VOC in Fig. 14

Model	Method	Datasets			
		COCO	PVOC	PContext	ADE20K
GroupViT	Original	24.3	52.29	22.39	8.56
GroupViT	Fine-tuning	25.64	48.26	21.33	**7.64**
GroupViT	**Fine-tuning with refined extraction**	**26.2**	**49.37**	**21.88**	7.61

Table 8: Fine-tuning with Refined Extraction of Labels. We compare vanilla fine-tuning with the model trained with refined label extraction methodology. We also provide mIoU score of the pretrained model for reference.

Model	Method	Datasets			
		COCO	PVOC	PContext	ADE20K
GroupViT	Original	24.3	52.29	22.39	8.56
	Fine-tuning	25.64	48.26	21.33	**7.64**
	Fine-tuning with Refined Text Set	**26.2**	**49.37**	**21.88**	7.61

Table 9: Fine-tuning with Refined Extraction of Labels. We compare vanilla fine-tuning with the model trained with refined label extraction methodology. We also provide mIoU score of the pretrained model for reference.

5.7.1 Extract Labels with Most Frequent Entities

In our text augmentation process, detailed in Section 4.3, we extract nouns from caption using NLTK. It is often observed that NLTK misidentifies nouns and extracts words like 'twice', 'exact', 'Jonas', 'TYPING', etc., despite our focus on Nouns ('NN'), Plural Nouns ('NNS'), and Proper Nouns ('NNP'). It is important to note that NLTK tags such as 'NN', 'NNS', and 'NNP' serve as the basis for GroupViT's noun extraction from captions.

For our next set of experiments, we extract all nouns and their frequencies from the MSCOCO training dataset, resulting in 27,089 nouns. We observe that a substantial number of these nouns lacked clear entity representation. To address this, we choose two sets: one with nouns appearing more than 1000 times (383 nouns) in the entire training dataset, and another with frequencies exceeding 5000 (88 nouns). Our intention was to create cleaner label sets that would offer a more precise entity representation.

During training, we extract nouns and retain those belonging to the selected noun sets. If the count of retained nouns fall below the desired number of labels, we resort to using other nouns extracted from captions using the NLTK library, following the standard procedure.

Intriguingly, we observe that the model exhibits better performance with the conventional extraction method. Our speculation revolves around the idea that selecting nouns based on their frequency might overly focus the model on a narrow selection of entities, hampering its ability to learn other entities or their synonyms.

The results of our experiments utilizing the curated set of nouns with frequencies exceeding 1000 and 5000 were shown in Table 10. However, since we do not observe significant performance improvements and witness less promising performance during

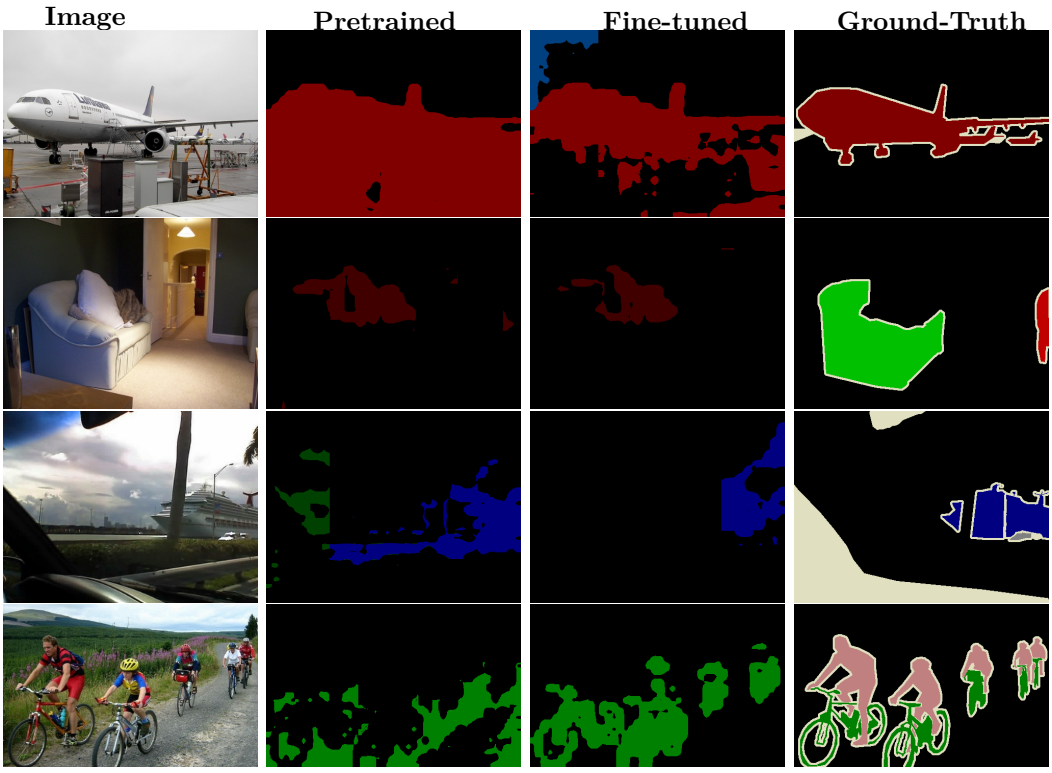


Figure 14: Qualitative Analysis after Fine-tuning GroupViT on Pascal VOC

the initial epochs, we restrict the model training to the initial 5 epochs.

5.8 Entropy Regularization

We now employ entropy regularization penalties thoroughly discussed in Section 4.7. We will conduct experiments with three penalties: (i) Penalty for Entropy over Groups, denoted as GE, detailed in Section 4.7.1, (ii) Penalty for Entropy over extracted labels, denoted as LE, detailed in Section 4.7.2, (iii) Entropy over Groups and Labels in the 'Segment Affinity Metric', denoted as SE, detailed in Section 4.7.3. Our experiments involve various combinations of these penalties as shown in Table 11. Our findings indicate that all the regularization methods perform better than the last baseline reported in Table 9. Moreover, we observe that there is no advantage in using combination of regularization methods. Overall, 'LE' outperforms rest of the methods with a slight margin. However, all these methods still falter on the downstream datasets. To further evaluate the efficacy of entropy regularization, we plan to test its impact at higher resolutions, as outlined in Section 5.9.1.

Model	Method	Datasets			
		COCO	PVOC	PContext	ADE20K
GroupViT	All Entities	**25.76**	**49.20**	**21.87**	**7.50**
GroupViT	Entity_freq>= $1K$	25.25	48.70	21.72	6.95
GroupViT	Entity_freq>= $5K$	23.86	46.87	21.19	6.63

Table 10: Impact of Curated Labels. We assess the impact of curated labels by comparing the mIoU scores of three models: one trained with labels having frequencies over 1K in the COCO Caption dataset ('Entity_freq≥ 1K'), another with labels having frequencies over 5K ('Entity_freq≥ 5K'), and the vanilla model, which includes all entities.

Model	Method	Datasets			
		COCO	PVOC	PContext	ADE20K
GroupViT	Original	24.3	**52.29**	22.39	**8.56**
GroupViT	Fine-tuned(RE)	26.2	49.37	21.88	7.61
GroupViT	SE	26.61	48.78	21.61	7.54
GroupViT	LE	**26.76**	49.69	21.92	7.53
GroupViT	GE	26.61	50.16	22.41	7.82
GroupViT	GE + SE	26.41	50.92	22.65	7.51
GroupViT	LE + SE	26.61	49.77	21.8	7.32
GroupViT	GE + LE	26.66	51.41	**22.67**	7.67
GroupViT	GE + LE + SE	26.61	50.67	22.47	7.72

Table 11: Entropy Regularization. Here, first row represents the original checkpoint. Fine-tuned(RE) refers to model fine-tuned with refined label extraction methodology. GE, SE and LE indicates models trained with penalty for entropy over groups, entropy over segment affinity metric, and entropy over labels respectively. '+' between them indicates model is trained with more than one loss function.

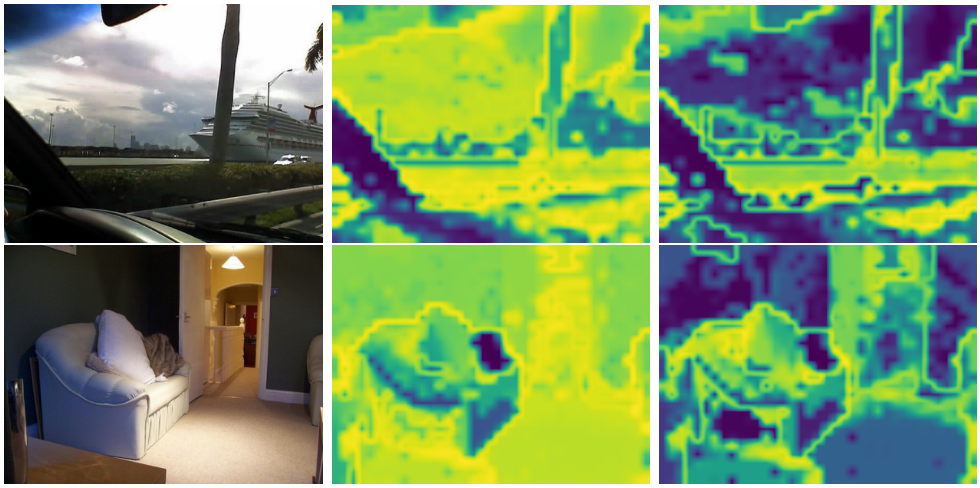


Figure 15: Visualization of Entropy Distribution: Left - Image. Center - Entropy of Patch tokens over Group Tokens: Pre-regularization. Right - Entropy of Patch tokens over Group Tokens: Post-regularization.

5.9 Fine-tuning on High Resolution

Dosovitskiy et al. propose fine-tuning Vision Transformers (ViT) at a higher resolution than the resolution used in pre-training[1]. This approach has demonstrated enhanced model performance, highlighting resolution's role in fine-tuning strategies. Therefore, we adopt a higher resolution of 384, deviating from the standard 224, for our following experiments.

To achieve this, we utilize positional encoding interpolation, similar to ViT[1]. Specifically, we employ bicubic interpolation to resize the positional encoding loaded from the pretrained model to the new dimension. Subsequently, we rearrange the tensor to match the expected format within the model and update the positional encoding parameter in the model's state dictionary with the appropriately resized positional encoding.

We present our findings in Table 12. Our experimentation unveiled a noteworthy insight: fine-tuning the model with an image resolution of 384 improved the training process. However, this improvement came at the expense of compromised performance on downstream datasets.

5.9.1 High Resolution Complements Entropy Regularization

Continuing our investigation, we explore higher resolution training with entropy regularization. For this, we use group entropy loss (GE), label entropy loss(LE)

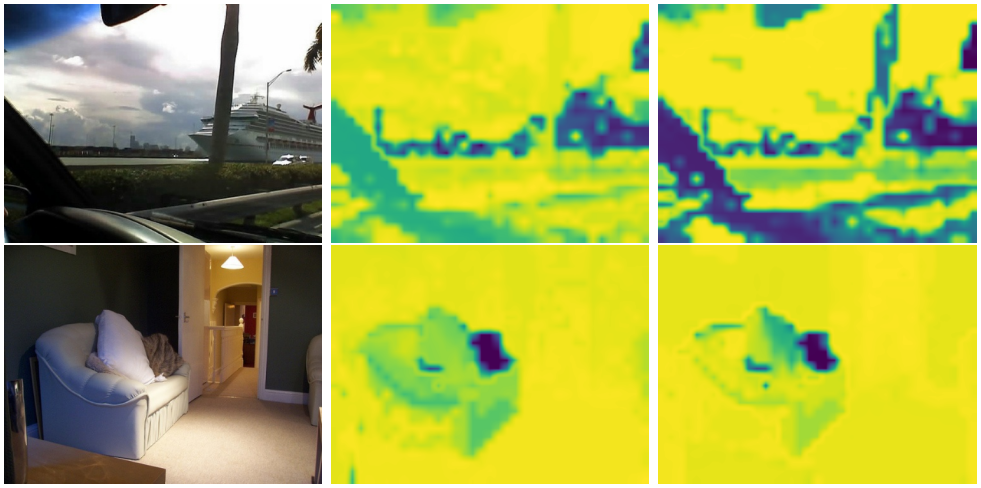


Figure 16: Visualization of Entropy Distribution over Labels: Left - Image. Center - Entropy of Patch tokens over Labels: Pre-regularization. Right - Entropy of Patch tokens over Labels: Post-regularization.

and penalty for entropy over segment affinity metric(SE). We present our findings in Table 12 which shows improved performance for 'GE' on COCO as well as on PASCAL VOC while performance on PASCAL Context and ADE20K remain steady. We observe some improvements for 'LE' and 'SE' on COCO but these methods still falter on the downstream datasets.

We also provide visualizations of entropy changes over groups and labels, depicted in Fig. 15 and 16, both pre- and post-regularization for two images from PASCAL VOC. Figure 15 highlights a significant reduction in pixel entropy over groups post-regularization for both the images. Conversely, in Fig. 16, we observe improvements in the top row but limited changes in the bottom row. This aligns with our observation that employing 'GE' is more beneficial than 'LE' specifically on downstream datasets. These samples can also be visualized in qualitative examples in Fig. 17

We note that training at 384 resolution increases resource requirements: an epoch takes about 7 more minutes than at 224 resolution, translating to 3.5 hours for 30 epochs. However, as visualized in Fig. 25b, the model shows significant speedup over previous baselines. In summary, fine-tuning on a higher resolution of 384 with entropy regularization improves the performance without impacting performance on the downstream datasets.

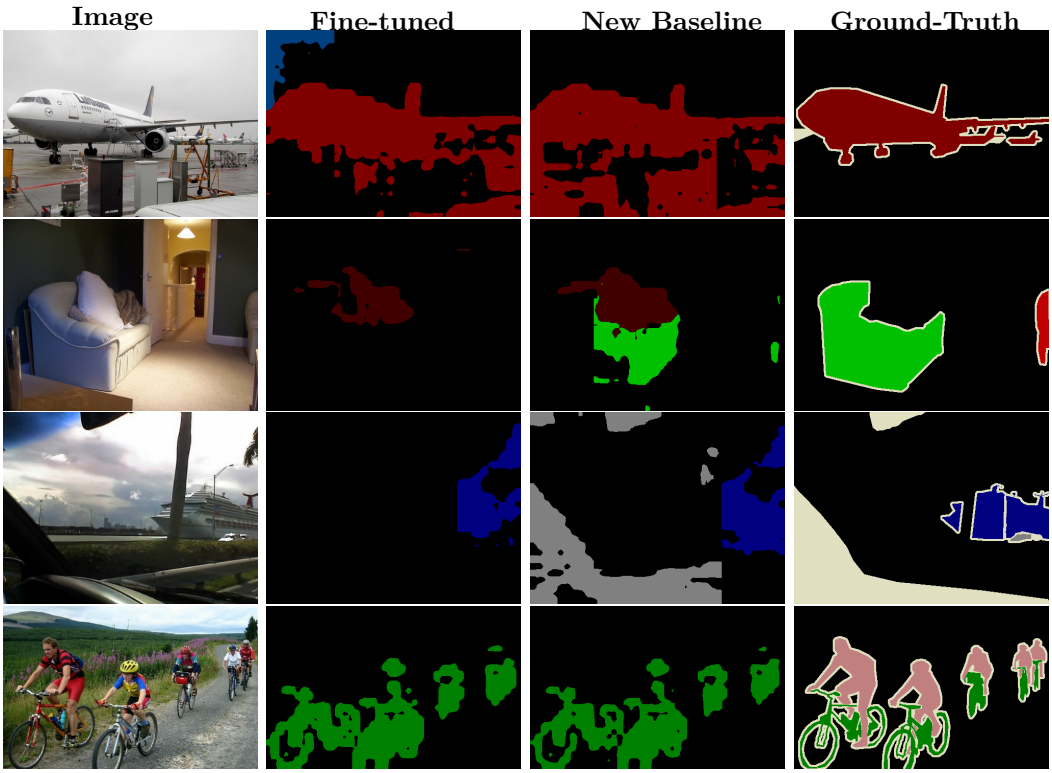


Figure 17: Qualitative Analysis on Pascal VOC for model trained with Entropy Regularization on a higher resolution of 384

5.10 Non-noisy Contrastive Loss

As described in Section 4.5.2, during the pretraining phase, GroupViT extracts labels from captions to create image-text pairs, which are then considered as positive instances within the multi-label loss formulation. For fine-tuning, we use COCO, comprising just 600,000 data points, introducing a challenge.

Specifically, there's a significant likelihood that same labels may appear across different images, potentially causing them to be misconstrued as negatives rather than their rightful status as positives.

To address this, we extend our strategy to include identical labels from other images within the same mini-batch as positives. Notably, these same labels might have been prompted differently for other images. To address this, we check the labels in their original string representation. Additionally, comparison of labels in their string representation is more viable and less time-consuming than comparing embeddings. As a result, we avoid using a global batch to create the negative pool, which would require combining embeddings from mini-batches across all distributed systems.

Method	Training Resolution	Datasets			
		COCO	PVOC	PContext	ADE20K
Original	224	24.3	52.29	**22.39**	**8.56**
Fine-tuned	384	26.81	46.9	20.82	7.77
SE	384	27.24	49.54	21.61	7.74
LE	384	27.25	49.46	21.87	7.74
GE	384	**27.86**	**53.38**	22.25	7.66

Table 12: High Resolution Fine-tuning. Here, Fine-tuned is the model trained with refined extraction. GE is the model trained with group entropy loss. LE is trained with label entropy loss. SE is the model trained with penalty for entropy over the 'segment affinity metric'.

Consequently, this leads to a reduction in the number of negative pair available for comparison. Nonetheless, we observe a noteworthy trend: prioritizing the accuracy of positive pairs, even if it comes at the expense of negative pool, correlates with improved overall performance.

Given the resource-intensive nature of this approach, we execute this experiment with a batch size of 128 across 4 machines, making a global batch size of 512. We provide our findings in Table: 13. We observe superior performance of this approach on validation split of COCO and downstream datasets. We provide a qualitative analysis on COCO, PASCAL VOC and PASCAL Context in Fig. 18. Additionally, we provide exhaustive qualitative analysis on individual datasets in Section 5.16

Model	Method	Datasets			
		COCO	PVOC	PContext	ADE20K
GroupViT	Original	24.3	52.29	22.39	**8.59**
GroupViT	Fine-tuning	26.81	46.9	20.82	7.77
GroupViT	GE	27.86	53.38	22.25	7.66
GroupViT	Non-noisy CL + GE	**28.68**	**53.41**	**23.34**	7.96

Table 13: Non-noisy Contrastive Loss. Here, Fine-tuning is fine-tuning with refined extraction. GE denotes Group Entropy Regularization baseline. Non-noisy CL + GE denotes model trained with noise-free contrastive loss and Group Entropy Regularization. All the models are trained on the resolution of 384.

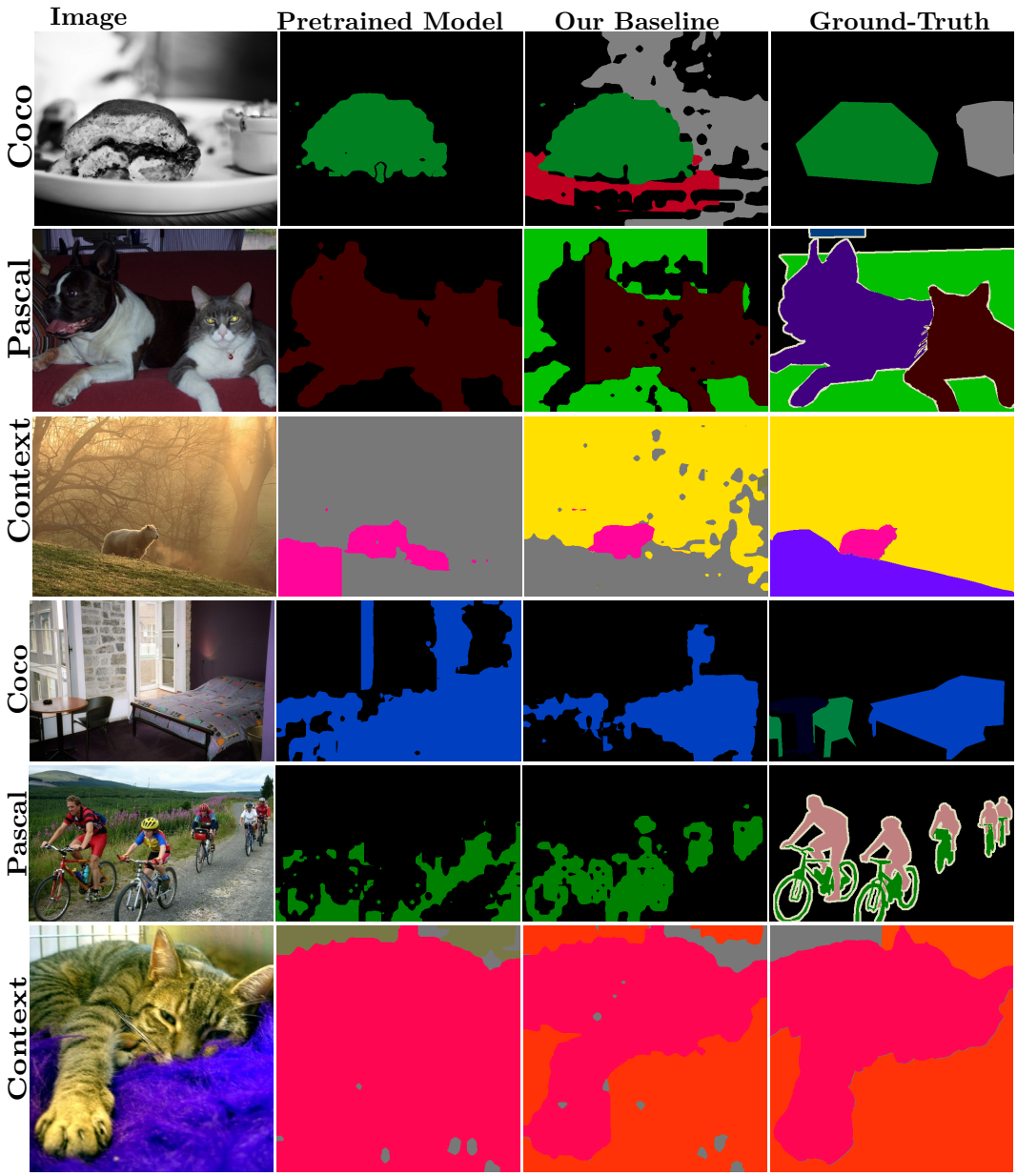


Figure 18: Qualitative analysis of the baseline across datasets. We visualize segmentation maps obtained from PASCAL VOC, COCO and PASCAL Context for the model trained with noise-free contrastive loss as reported in Table 13.

5.11 Impact of Inference Settings

Next, we analyze how inference parameters like resolution and background threshold influence the result. We explore impact of inference resolution and background threshold. The background threshold influences background pixel recognition: lower values enhance foreground coverage, while higher values require more confidence for foreground assignment. This parameter's tuning balances false positives and negatives, refining segmentation accuracy. Our results are in Table 14, where we show that for COCO a resolution of 512 with a background threshold of 0.95 works better than other settings. However, for PASCAL VOC , a really high threshold of 0.99 works the best.

Model	Inference Res	Bg Threshold	Datasets	
			COCO	PVOC
GroupViT	448	0.9	28.68	52.03
GroupViT	448	0.95	28.71	53.41
GroupViT	448	0.98	28.44	54.89
GroupViT	448	0.99	28.43	55.07
GroupViT	512	0.9	28.61	53.05
GroupViT	512	0.95	**28.96**	54.21
GroupViT	512	0.98	28.9	55.31
GroupViT	512	0.99	28.89	**56.41**

Table 14: Ablation on Inference Parameters. Here, 'Inference Res' represents resolution of images set during inference and 'Bg Threshold' represents Background Threshold.

5.12 Analysis of The Baseline

5.12.1 Visualization of IoU Disparity

In this section, we delve into a comprehensive analysis of our baseline model, focusing on visualizing its impact on individual classes. To achieve this, we calculate the Intersection over Union (IoU) disparity across classes for the COCO dataset, as depicted in Figure 20. Given the large number of classes, we present IoU disparity for the baseline against the pretrained model for a selection of 20 classes: the top-10

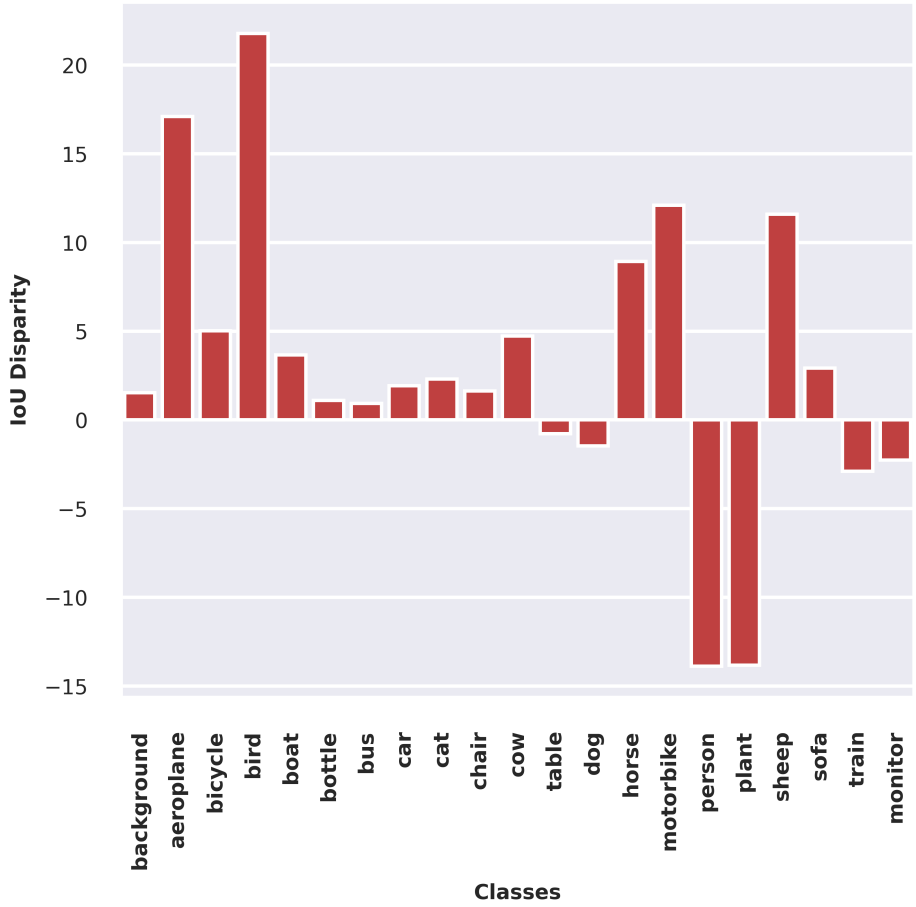


Figure 19: IoU Disparity per Class Compared to Pretrained Model on Pascal VOC Dataset

and bottom-10 ranked by IoU disparity.

Our observations reveal interesting insights. Classes such as 'bench', 'scissors', and 'sink' exhibit notably lower IoU compared to the pretrained model, with classes 'person', 'train', and 'traffic light' also displaying diminishing performance. On the other hand, categories like 'baseball glove' and several categories representing animal kingdom show significant improvement. To gain a deeper understanding, we offer a qualitative view of classes with low mIoU in Figure 22.

In our qualitative analysis, we notice recurrent issues. For instance, 'person' is frequently mislabeled as surrounding objects like 'tie' or 'baseball glove'. This trend is visible across various instances, illustrated in the initial rows of Figure 22. Additionally, over-masking tendencies arise in other classes like 'traffic light' and

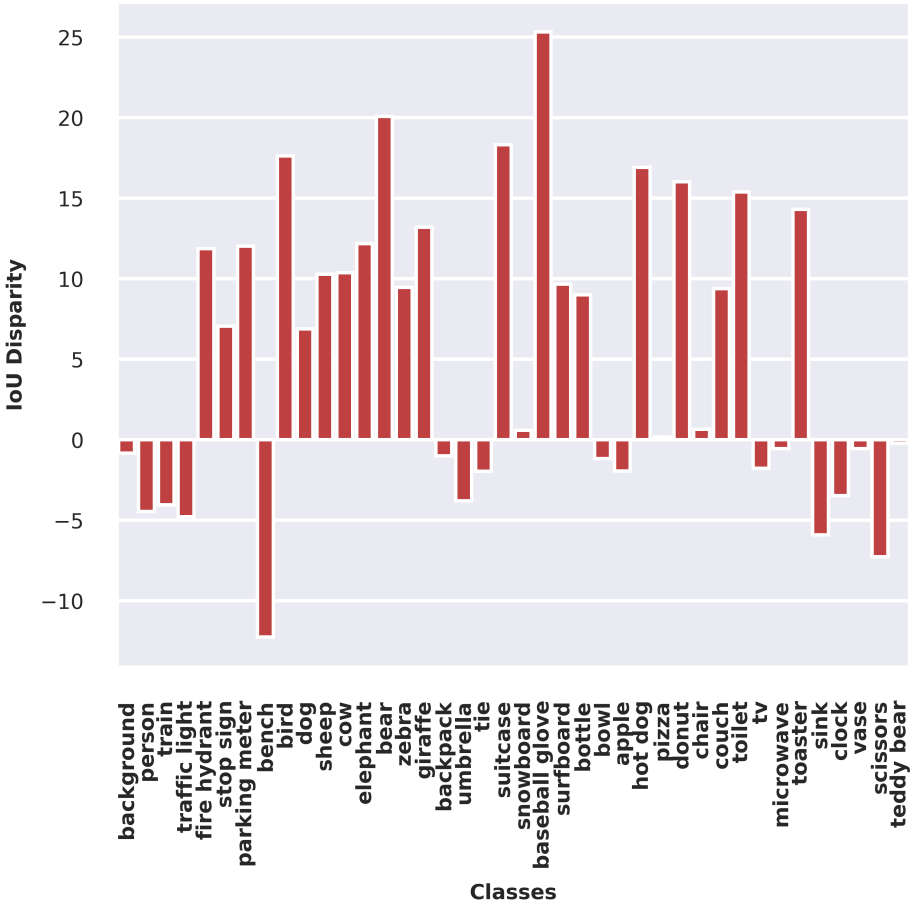


Figure 20: IoU Disparity per Class Compared to Pretrained Model on COCO Dataset. We depict IoU differences for the 20 classes with positive and negative deviations from the pretrained model on COCO Dataset.

'sink' where the model masks object and the associated structure together. Due to GroupViT's reliance on text-based weak supervision, the model struggles to attain pixel-level precision in segmentation. It tends to associate common object-related structures with the object itself, thus leading to lower mean IoU (mIoU) scores.

After fine-tuning on MSCOCO, the model demonstrates improved performance on dataset-specific classes, such as 'surfboard', 'basketball glove', and 'hotdog'. However, a recurrent issue persists — humans in the vicinity are occasionally mislabeled as these specific object classes due to their association with these items. This exemplifies the challenges arising from weak supervisory signals.

Moving on to the Pascal VOC dataset (PASCAL VOC), we proceed to compute the IoU disparity in relation to the pretrained model, visualized in Fig. 19. Notable enhancements emerge for classes like 'aeroplane', 'bicycle', 'motorbike', 'horse', and 'sheep'. However, certain classes such as 'person' and 'plant' exhibit poorer performance. To gain deeper insights, we present a qualitative analysis of these classes in Figure 23.

Specifically, in the case of 'plant' and 'person', which show higher negative IoU disparity, we visually explore the reasoning through the initial rows of Figure 23. When 'plant' is in the foreground, the model performs well, yet it mislabels 'trees' as 'plant' in second row. Additionally, when 'plant' is not the prominent object—as in the third row—it often remains unidentified. Similarly, the 'person' mislabeling issue seen in COCO resurfaces here as well.

In the context of other structures, such as 'monitor' and 'train', we observe over-masking tendencies. These structures tend to blend with their surroundings, resulting in the visualizations presented in Figure 23.

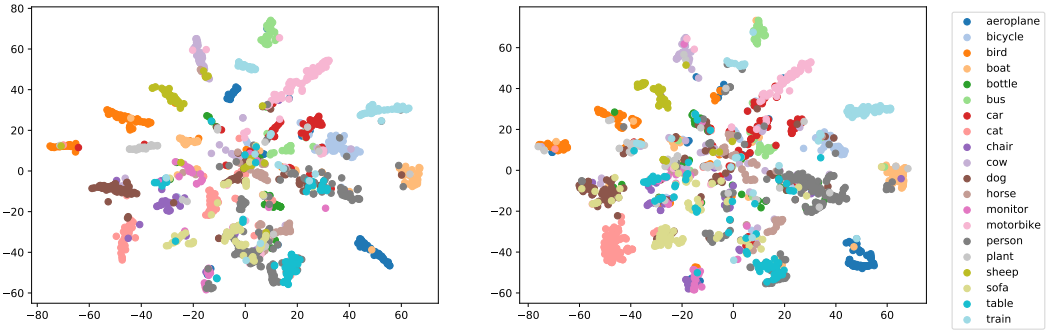
5.12.2 Visualization of Representation Space

To visualize the representation spaces created by our baseline, we employ t-SNE[50], a dimensionality reduction technique. To achieve this, we utilize the validation splits of both PASCAL VOC and COCO datasets. We extract segment token features from the final stage that our baseline and pretrained model label with respective dataset categories.

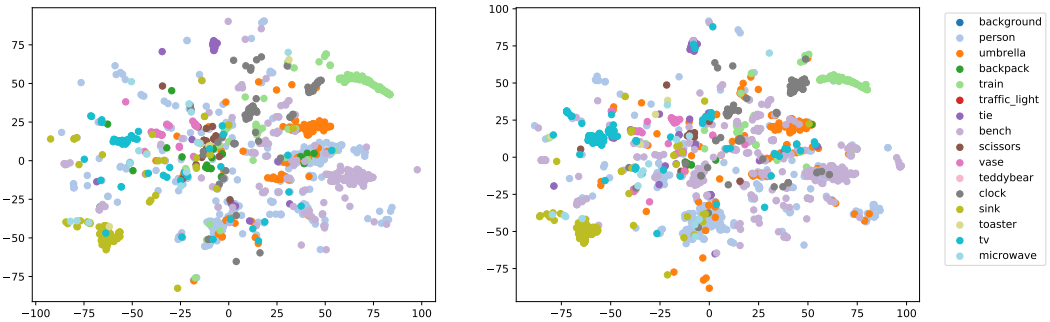
The outcomes are presented in Figure 21a and Figure 21b. Given COCO's extensive class variety, we concentrate solely on classes with negative IoU disparity, as shown in Figure 20. Our findings mirror those from the prior observations in Figure 20.

We note several trends for PASCAL VOC . For the cluster of 'plant', there's increased dispersion. Additionally, features corresponding to 'person' often appear near clusters associated with 'sofa' and 'table'. The 'monitor' cluster lacks clear demarcation from 'chair' and 'table'. Conversely, we observe improved clustering for 'aeroplane', 'motorbike', and 'sheep' features. These observations reinforce our earlier insights in Figure 19.

Our findings for COCO datasets also align with the insights from Figure 20. We observe that the features corresponding to 'bench' and 'person' do not converge, indicating mislabeling. Further, our observation resonates with the depiction in Figure 22, second row, where features of 'person' was identified as 'tie'. Additionally, 'scissors' and 'vase' have only few features, suggesting that the model still needs improvement



(a) Visualizations of learned representation space of Group Tokens for PASCAL VOC Left: features extracted from Group Tokens in pretrained model. Right: features extracted from Group Tokens in our baseline.



(b) Visualizations of learned representation space of Group Tokens for COCO classes with low IoU. Left: features extracted from Group Tokens in pretrained model. Right: features extracted from Group Tokens in our baseline.

Figure 21: Visualizations of Learned Representation Space

in identifying and grouping these classes. We hypothesize that these classes either frequently coexist with multiple objects or occupy relatively smaller visual areas. Once again, this underscores the challenge of achieving pixel-level accuracy due to the constraints of weak supervision, particularly in scenarios involving multiple objects within a single visual context.

5.13 Exploring Number of Stages in Visual Encoder

As we observed a bigger room of improvement in performance of visual grouping in GroupViT demonstrated in Sec. 5.4, we further explore different number of stages of the GroupViT architecture.

Figure 22: Qualitative analysis on COCO Classes with low IoU. For the 'person' class, we see mistakes like labeling it as 'baseball bat' and 'tie' in the first two rows. In the third and fourth row, the 'person' is not classified. Moreover, in fourth row, the background is wrongly tagged as a 'bench'. Despite correctly identifying 'clock', it's being overmasked. Structures associated with 'Traffic light' and 'Sink' are also masked with the object in last two rows.

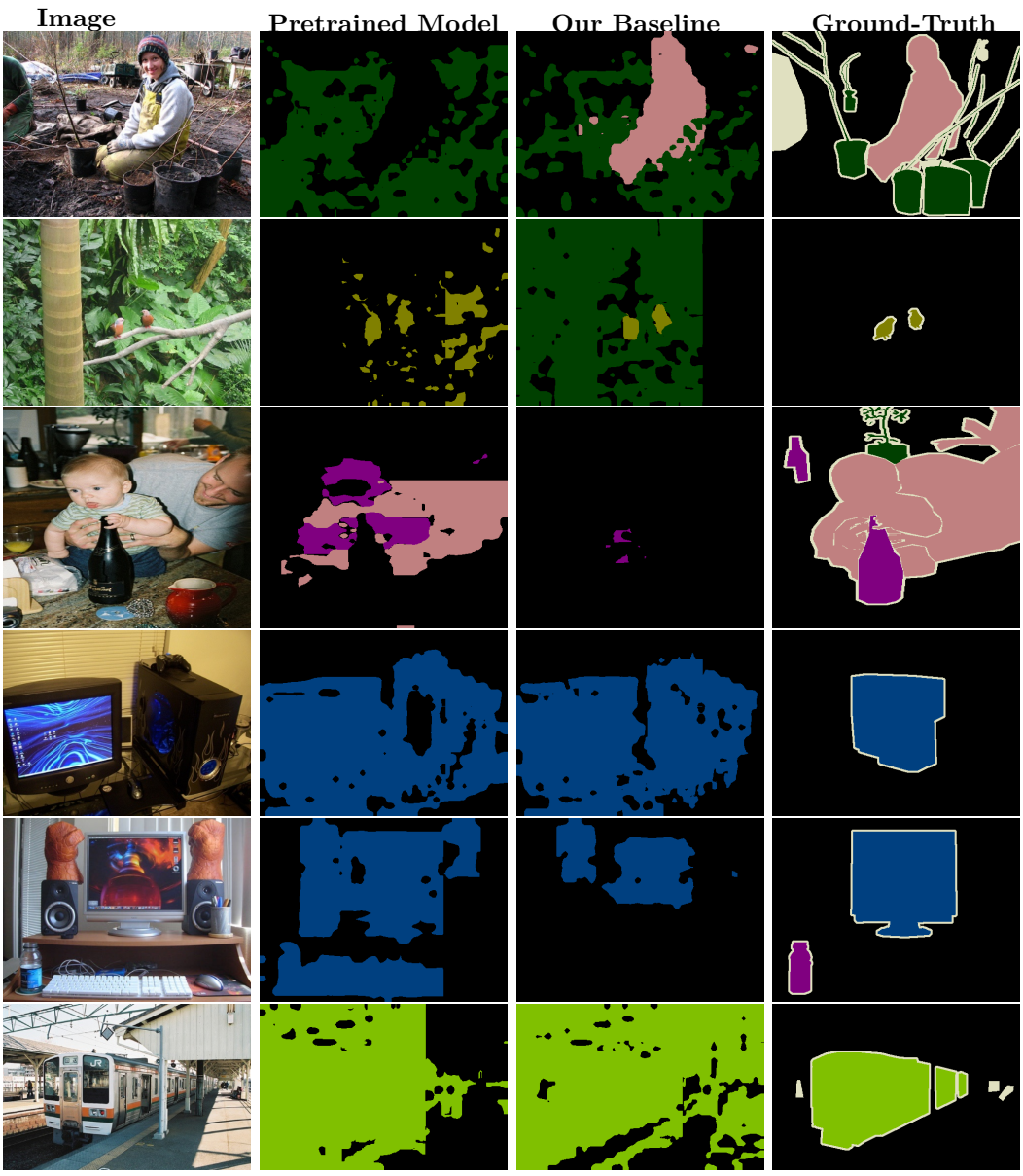


Figure 23: Qualitative analysis on PASCAL VOC Classes with low IoU. We observe that in the first row, 'plant' refines the mask compared to the pretrained model, but in the second row, it mistakenly identifies a tree as 'plant'. In the third row, the model struggles with precise classification of 'person' and 'bottle'. Finally, in the last row, we see the mask of 'train' incorrectly incorporating its associated structure.

5.13.1 Single Grouping Block

For single Grouping Block architecture, we retain only the first Grouping Block from the original design, removing the Grouping Block of stage2. The rest of the architecture remains unchanged, following the configuration of the original model. As the authors of GroupViT, Xu et al., did not provide a checkpoint for this architecture, we follow below steps to load the pretrained model's weights:

1. **Load Transformer Layer for Stage 1**: Load all the parameters till first stage from the original architecture, i.e. of 6 Transformer Blocks in stage 1 and the Linear Projector.

2. **Parameter-Name Mapping for Stage 2**: We map the parameter names of Transformer blocks 7 to 9 from stage 2 and Transformer blocks 10 to 12 from the final stage in the original checkpoint to Transformer blocks 6 to 12 in stage 2 of the new architecture and load their weights.

By following these steps, we make the new architecture capable of using weights from the pretrained model. This method was chosen to closely match the intended design and behavior.

For a 10-epoch training session on our model, we record the results in Table 15. The changes in architecture lead to a reduced retention of knowledge from the pretrained model. Consequently, the initial inference results in a very low mIoU value on MSCOCO. Recognizing this knowledge loss, we train all parameters of the visual encoder. Although there's some improvement in later epochs, it's not significant.

Despite attempting to improve performance through learning rate optimization, we found that progress was limited. The significant loss of information resulting from architectural changes highlights the need for a resource-intensive web-scaled dataset like YFCC and CC12M, similar to the approach taken by Xu et al. in training GroupViT. Unfortunately, resource constraints prevented us from pursuing this path.

5.13.2 Three Grouping Blocks

In the next step, we enhance the GroupViT architecture by adding one more Grouping Block, resulting in a total of three Grouping Blocks. These Grouping Blocks are positioned after the 4th, 6th, and 9th Transformer blocks, which differs from the original model. We then perform two experiments: (i) keeping 8 group tokens in the final stage and (ii) reducing group tokens to 4 in the final stage to encourage focused convergence.

Integrating pretrained weights poses challenges, prompting us to develop a weight loading scheme similar to what was discussed in Section 5.13.1. Specifically, we load the first four transformer blocks' weights as they are. For the remaining architecture, we employ a parameter-name mapping scheme from the pretrained model to the new architecture, as described in Section 5.13.1, and then load corresponding weights.

While the count of Grouping Blocks has changed, the count of Transformer Blocks remains constant between the two architectures. After loading the first four Transformer blocks' weights directly, the remaining blocks follow a sequential process of mapping parameter names from the pretrained model to the new architecture. This ensures compatibility between the two versions. For the third Grouping Block, we load the weights from the second Grouping Block. For the second experiment with the Grouping Block with 4 Group Tokens, we load the weights of first 4 Group Tokens from the second Grouping Block in the original model

Despite these efforts, similar to what was observed in Section 5.13.1, we find that the model doesn't retain its pretrained knowledge effectively. We train all parameters of GroupViT for three stages, using two GeForce RTX3090 GPUs. Our findings, reported in table 15 for 20 epochs reveal poor performance.

We observe the initial inference scores of the models on the COCO object dataset are disappointingly low (mIoU < 1), aligning with the observations from Section 5.13.1. This suggests that the transfer of pretrained knowledge is imperfect due to architectural adjustments. Since resource limitations prevent a thorough stage-wise analysis, we are unable to further pursue this experiment due to computational demands.

Model	#Grouping Blocks	#Group Tokens	Datasets			
			COCO	PVOC	PContext	ADE20K
GroupViT	2	8	**25.96**	**48.67**	**21.69**	**7.60**
GroupViT	1	64	13.5	29.7	0.26	0.03
GroupViT	3	8	13.5	11.68	1.04	0.31
GroupViT	3	4	9.5	20.55	1.55	0.75

Table 15: Exploring hierarchy of Visual Encoder. We calculate mIoU scores for various model architectures, which vary in the number of Grouping Blocks and Group Tokens in the final stage.

5.14 DINO Features

A recent study, by Amir, Shir, et al. [2], explored patch-level features extracted from the DINO model's dense layers. These features proved highly effective in tasks like part co-segmentation and semantic correspondences, particularly for capturing fine-grained spatial details. The authors used lightweight techniques such as clustering and binning to leverage the various representations available for each patch, including query, key, value, and token forms.

One important discovery was the exceptional performance of the key facet from layer '9' in encoding strong representations for semantically coherent concepts. In this section, we first compare the grouping of coherent concepts by GroupViT and DINO and then attempt to use these features to enhance the grouping capabilities of GroupViT

5.14.1 Feature Affinity Metric

First, we analyze features extracted from both models to assess their suitability for grouping semantic concepts together.

In a study conducted by Melas-Kyriazi, Luke, et al., the authors investigate the eigenvectors of the Laplacian derived from a feature affinity metric. This feature affinity metric is computed by calculating the cosine similarity between each pair of patch token features. They discover that these eigenvectors can decompose an image into meaningful segments, making them suitable for tasks like object localization and segmentation [17]. This finding suggests a connection between the feature affinity metric and the grouping of coherent concepts.

To assess the efficiency of visual grouping for both DINO and GroupViT, we calculate the feature affinity metric and compare them. For this analysis, we obtain $\mathbf{D_f} \in \mathbb{R}^{P \times D_D}$ from DINO, where P represents the number of patches. In the case of GroupViT, we acquire features after the first stage as they consist of the same number of tokens as in DINO. These features are denoted as $\hat{\mathbf{S}}^1 \in \mathbb{R}^{|S^1| \times D}$, as described in Eq. 16. Here, $|S^1|$ is equal to P, and D_D is equal to D, signifying the dimensions of both models. Following this, we compute the cosine similarity between each patch token and every other patch token within their respective feature spaces in their respective models.

Our hypothesis is that patch tokens expected to exhibit similarity should display strong similarity, while tokens anticipated to be dissimilar should reflect their dissimilarity strongly. We visually depict these feature affinity patterns in Fig. 24,

selecting the first 10 images from the validation split of the PASCAL VOC 2012 dataset. The visualizations consistently show that DINO strongly expresses similarity or dissimilarity with its associated patch tokens, distinguished by blue and red colors. In contrast, similarities in GroupViT's feature affinity metric are less pronounced, typically clustering around mid-range values on the scale. These comparative scales are visually demonstrated for each metric in Fig. 24.

Since the feature affinity metric has a strong correlation with the grouping of coherent concepts, and DINO demonstrates stronger performance by showing stronger similarities and dissimilarities, we aim to distill the knowledge of this unsupervised method into GroupViT.

5.14.2 Integration of DINO Features in GroupViT

We first aim to analyze if Grouping Blocks of GroupViT can be readily used to group DINO features into segments as previous works have already demonstrated suitability of DINO features to group coherent concepts just by employing light-weight methods such as clustering, binning, spectral clustering [2][17]. We introduce DINO features $\mathbf{D_f}$ as input in Stage 2. This involves utilizing $\mathbf{D_f}$ and $\mathbf{G}^2$ as inputs to be grouped by Grouping Block of second stage. We observe a low mIoU of 1.28 on COCO, leading us to abandon this method as well.

We hypothesize that Grouping Blocks pretrained to group GroupViT's features cannot be readily used to group DINO features. Therefore, we shifted our approach towards leveraging a loss function to distill knowledge acquired by DINO.

5.14.3 Distillation of Information from DINO to GroupViT

We attempt to distill the information of DINO features in GroupViT model with an aim to improve visual grouping. To do so, we treat DINO as the Teacher model and GroupViT as the Student model. Specifically, we employ a cross-entropy loss with feature affinity metric of DINO model as the target. For this, We first calculate feature affinity metric from DINO features. Then, We calculate row-wise softmax on the obtained feature affinity metric, as shown in Eq. 54

Next, we obtain global soft attention, $\mathbf{Attn} \in \mathbb{R}^{|G^2| \times |S^1|}$, denoted in Eq. 34. We then perform dot product of $\mathbf{Attn}$ with $\mathbf{Attn}^T$ to obtain a similarity metric of the same dimension as DINO: 196×196, shown in Eq. 54. We show the formulation of this loss in Eq. 54

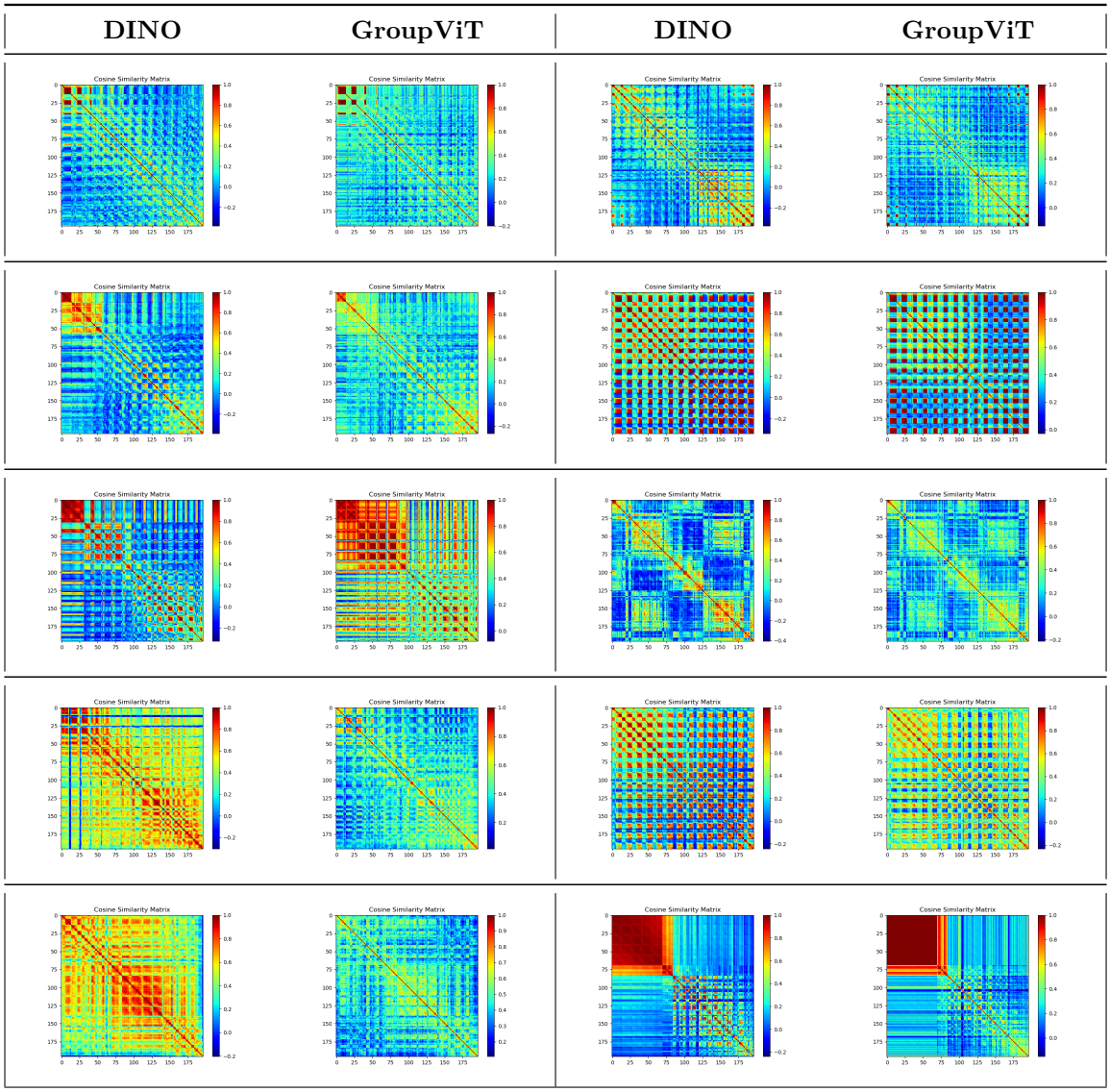


Figure 24: DINO vs GroupViT: Patch Feature Affinity Metric. We take patch features from DINO layer 9 and GroupViT after the first stage. We then calculate an affinity metric between these patch token features using cosine similarity. We visualize the obtained metrics for DINO-GroupViT pair for 10 images from the PASCAL VOC validation dataset.

$$\mathcal{L}ce = -\sum_{i=1}^{P}\sum_{j=1}^{P}(\text{Softmax}(\mathbf{DINOSim}))_{i,j} \cdot \log((\mathbf{Attn} \cdot \mathbf{Attn}^T)_{i,j}) \tag{54}$$

We conduct this experiment for 10 epochs while keeping the settings unchanged. The result is shown in Table 16. Interestingly, we did not observe any notable improvement after employing the loss function mentioned in Eq. 54. However, we acknowledge that the process of distilling visual grouping insights from DINO to GroupViT remains an open question. Considering the diverse and extensive possibilities in this area, we believe this holds promise for future research.

Model	Datasets			
	COCO	PVOC	PContext	ADE20K
GroupViT + DINO	24.21	45.97	20.26	7.14
GroupViT	**25.96**	**48.67**	**21.64**	**7.60**

Table 16: GroupViT with DINO Features. Here, we report mIoU for GroupViT model trained with refined extraction as in Section 5.7 and model additionally employing DINO distillation loss.

5.15 Visualization of Curves

In this section, we provide an overview of our baseline approach and present visualizations of validation curves for key steps. We also investigate the impact of using multi-label contrastive loss in addition to contrastive loss, as discussed in Section 5.6. For this experiment, we conduct three runs with different seeds and show the mean results along with their variance in Figure 25d. While the model trained with multi-label contrastive loss shows similar performance to the model trained only with contrastive loss in later stages, we do observe a slight initial improvement when both losses are used. Therefore, we decide to adopt the multi-label strategy, aiming to explore its potential for further improvement, in line with the findings from Xu et al.'s GroupViT training [49].

Moving forward, in Figure 25a, we illustrate a significant improvement achieved by fine-tuning with refined text extraction compared to standard fine-tuning, as discussed in Section 5.7. Next, we delve into the integration of entropy regularization techniques, as described in Section 4.7. Interestingly, we observe that using high resolution in combination with entropy regularization outperforms all previous baselines.

Furthermore, we introduce the noise-free contrastive loss, as explained in Section 5.10, which serves as our final baseline. This modification results in an enhancement of 18

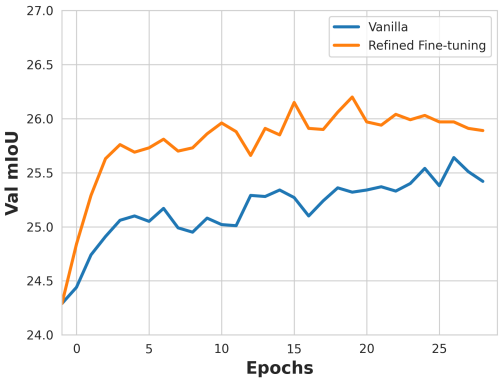
5.16 Qualitative Analysis

In this section, we perform the qualitative analysis of our baseline, which is trained using the noise-free contrastive loss as outlined in Section 5.10. This analysis is visualized in Figure 26. Notably, we observe improved model performance in both grouping and recognition aspects. Enhanced grouping is evident in rows 1 and 4, while mask refinement is observed in row 2. Additionally, the model demonstrates successful recognition of new classes, as seen in row 6.

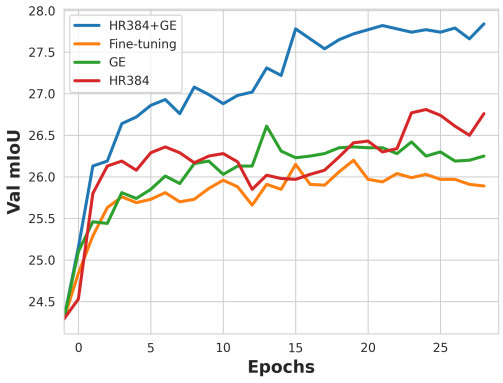
For the Pascal VOC dataset (PASCAL VOC), similar observations are made as depicted in Figure 27. Improved mask refinement and grouping are noted in rows 2, 4, and 6, where new concepts such as 'person', 'bottle', and 'sofa' emerge. However, a mislabeling of 'dog' as 'cat' is still observed in row 6.

Moving on to the PASCAL-Context dataset (PASCAL Context), we find that the model encounters challenges in grouping and labeling background classes like 'wall', 'ground', and 'sky'. This is unsurprising given the limited presence of such classes in the COCO dataset. Nevertheless, improved grouping relative to the pretrained model is still discernible.

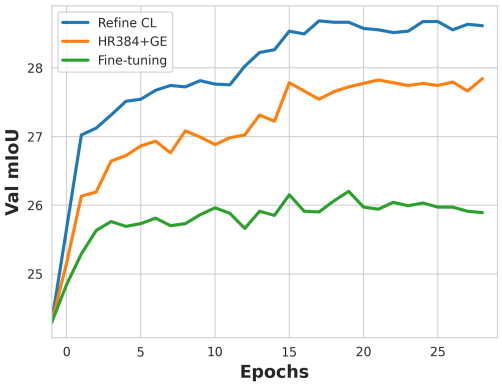
In summary, this section provides a detailed visual analysis of our baseline approach, shedding light on its performance across various datasets and classes.



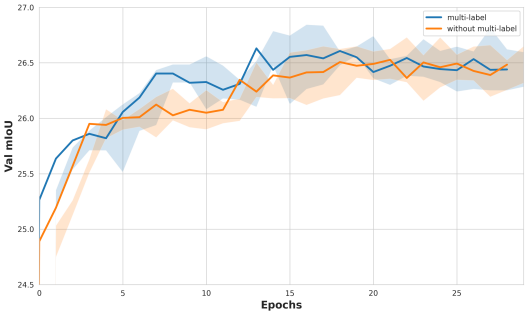
(a) Vanilla Fine-tuning vs Refined Fine-tuning



(b) High-resolution complements Entropy Regularization



(c) Noise-free Contrastive Loss



(d) Multi-Label Vs. Without Multi-Label

Figure 25: Visualization of validation curves of different baselines. Top-Left: we observe fine-tuning with refined text extraction, denoted as 'Refined Fine-tuning' performs better than vanilla fine-tuning, denoted as 'Vanilla'. Top-Right: We observe only marginal improvement with Group Entropy regularization, denoted as 'GE' and fine-tuning on a High Resolution of 384, denoted as 'HR384'. However when the model leverages both, denoted as 'HR384+GE', it outperforms all baselines. Bottom-Left: We compare 'Fine-tuning' (With refined label extraction) and 'HR384+GE' and the new baseline trained on Non-Noisy Contrastive Loss, denoted as 'Refine CL'. Bottom-Right: We compare the model trained with multi-label contrastive loss, denoted as 'multi-label' versus the one trained on both contrastive loss and multi-label contrastive loss, denoted as 'without multi-label'.

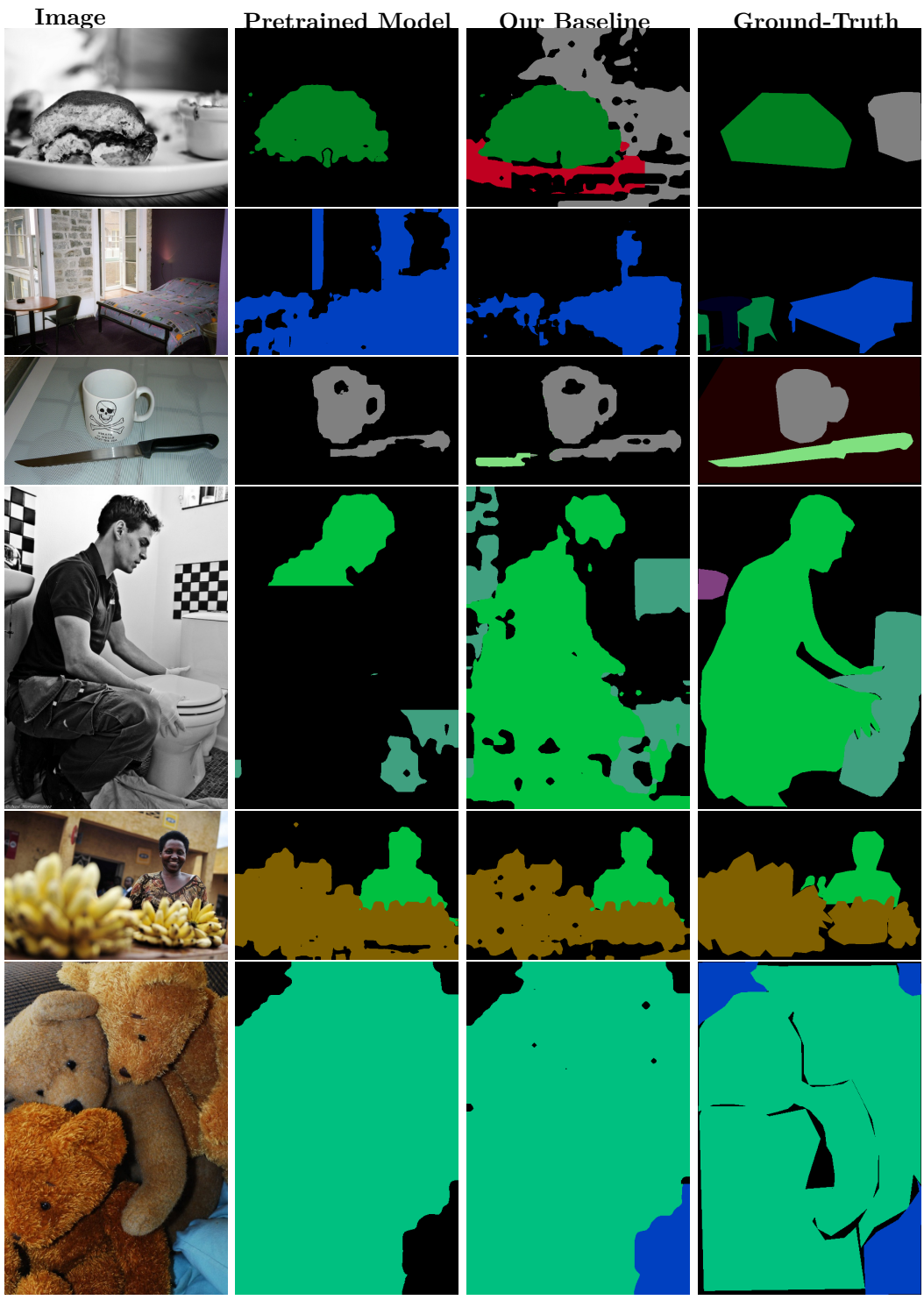


Figure 26: Qualitative analysis on COCO

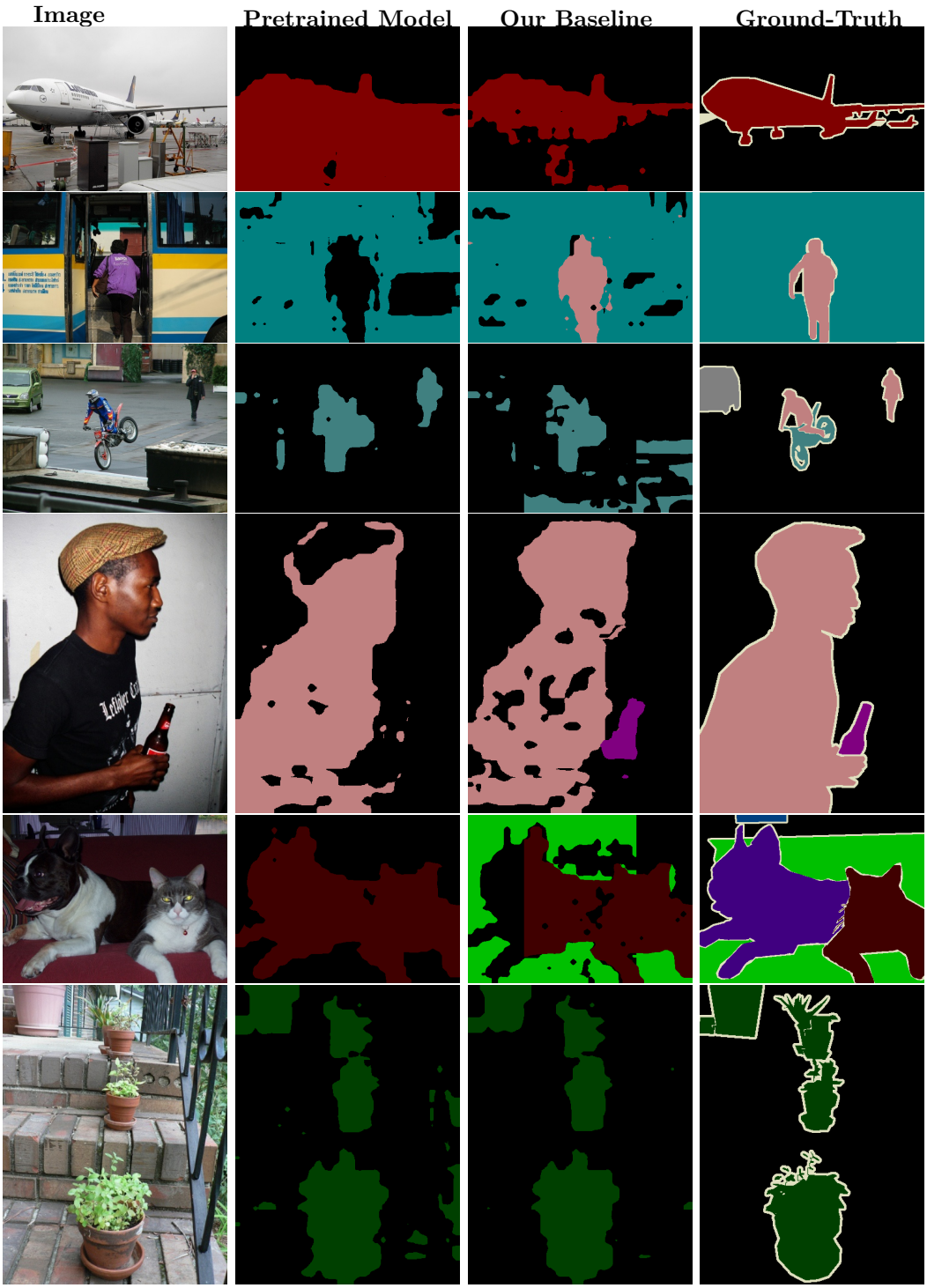


Figure 27: Qualitative analysis on PASCAL

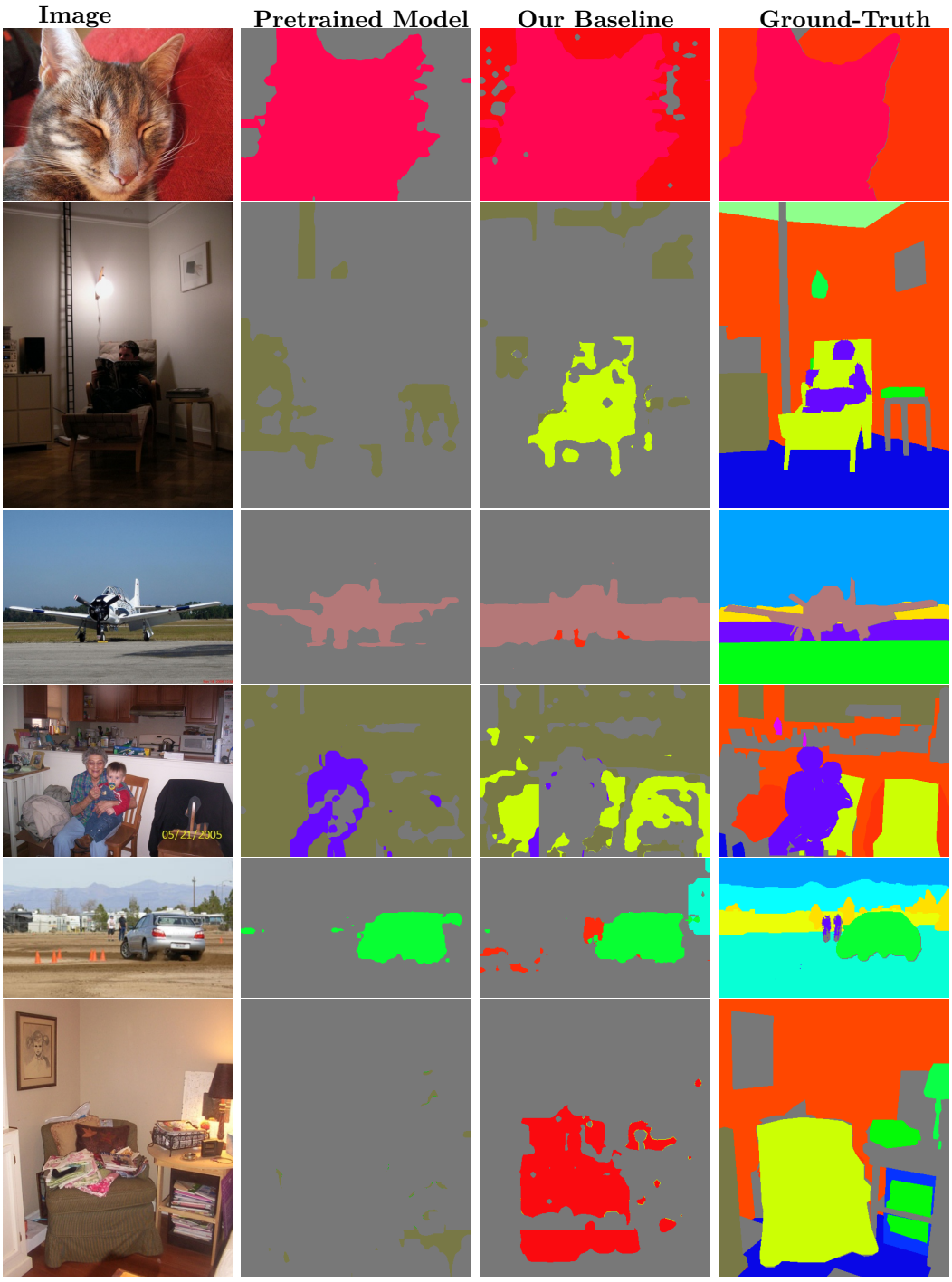


Figure 28: Qualitative analysis on Pascal Context

6 Conclusion and Future Work

GroupViT introduced a fresh perspective to semantic segmentation through a bottom-up approach. We closely examined GroupViT to grasp its two fundamental ideas: Grouping and Recognition. We then assessed each of these concepts individually to identify opportunities for improvement. We systematically pursued these enhancements to develop an efficient fine-tuning strategy.

During our investigation, we thoroughly analyzed various aspects of training and refining the text extraction process. To enhance our approach, we further introduced entropy regularization techniques. When our model is trained on high-resolution images, these techniques not only improve mIoU on the training dataset but also have a positive impact on downstream dataset performance.

In our pursuit of enhancing the model's performance further, we integrated a noise-free contrastive loss tailored for training on a cleaner, smaller dataset like MSCOCO. By following this recipe of fine-tuning, we witnessed an increase of 18% mIoU on the COCO dataset, all while maintaining consistent performance on other datasets such as PASCAL VOC and PASCAL Context.

Furthermore, we investigated the potential of utilizing GroupViT's grouping mechanism for grouping DINO features. In addition, we made progress in the direction of knowledge distillation from DINO to GroupViT by introducing a formulation, establishing a foundation for potential advancements in this area.

In essence, our exploration of GroupViT has involved dissecting its components, refining its training process while maintaining its capabilities on downstream datasets, and delving into opportunities for knowledge distillation. This journey has not only revealed the intricacies of GroupViT but also pointed toward promising directions for future advancements.

6.1 Future Work

The future possibilities for advancing GroupViT's capabilities present an exciting journey. We provide a brief overview of these prospects here:

6.1.1 Integration of DINO Features: Synergizing Strengths

A promising direction for future exploration involves integrating DINO and GroupViT. This integration would involve developing a unified framework to compare deep-layer features extracted from DINO and segment features extracted from GroupViT within a shared context. Additionally, investigating the potential of knowledge distillation could unveil new opportunities for enhancing the model's visual grouping capabilities..

6.1.2 Enriching Negative Sample Pool for Noise-Free Contrastive Loss

While the advantages of noise-free contrastive loss are clear, we must address the challenge of a limited negative sample pool. An effective approach to overcome this challenge is the creation of an extensive memory bank for negative samples. By this, the model can benefit from noise-free contrastive loss while maintaining a diverse set of negative samples, potentially resulting in enhanced performance.

6.1.3 Exploring Hierarchical Grouping with Web-Scaled Data

The potential of hierarchical grouping at more advanced stages remains unexplored due to resource limitations in our study. Expanding this research to include web-scaled datasets could provide valuable insights into the impact of hierarchical grouping on advanced stages of GroupViT.

6.1.4 Training with Datasets Including Stuff Classes

While GroupViT performs well on datasets like Pascal VOC and MSCOCO, it encounters difficulties with more complex datasets such as Context and ADE20K. These challenges arise from the presence of stuff classes like 'air', 'water', and 'sky', which are relatively underrepresented in human-annotated datasets like MSCOCO. To tackle this issue, future efforts could involve training GroupViT on datasets explicitly designed to include these stuff classes or specialized datasets created for this purpose. This approach has the potential to enhance GroupViT's adaptability to real-world scenarios and improve its performance across diverse scenes.

7 Acknowledgments

I extend my heartfelt gratitude to Professor Thomas Brox for his invaluable examination of this research thesis. His insightful feedback and guidance have played a pivotal role in shaping the outcome of this work. Additionally, I would like to express my sincere appreciation to Prof. Frank Hutter for sparking my interest in deep learning at the onset of my master's course and for examining my thesis.

I am deeply thankful for the unwavering support, mentorship, and expertise provided by my supervisors, Maria Bravo and Silvio Galesso, throughout the entire research process. Their continuous encouragement has been instrumental in driving my progress and enhancing the quality of this thesis.

The presence of my friends greatly enriched my research journey. Our collaborative sessions, shared experiences, and camaraderie played a pivotal role in shaping this meaningful academic endeavor. I want to extend my special thanks to Harsha for always being there with an open ear. Lastly, I am immensely grateful for my family's unwavering support; their belief in me and their love fueled my determination to achieve this significant academic milestone.

Bibliography

[1] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[2] S. Amir, Y. Gandelsman, S. Bagon, and T. Dekel, "Deep vit features as dense visual descriptors," *arXiv preprint arXiv:2112.05814*, vol. 2, no. 3, p. 4, 2021.

[3] X. Wang, R. Girdhar, S. X. Yu, and I. Misra, "Cut and learn for unsupervised object detection and instance segmentation," 2023.

[4] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9650–9660, 2021.

[5] Y. Wang, X. Shen, Y. Yuan, Y. Du, M. Li, S. X. Hu, J. L. Crowley, and D. Vaufreydaz, "Tokencut: Segmenting objects in images and videos with self-supervised transformer and normalized cut," *arXiv preprint arXiv:2209.00383*, 2022.

[6] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," 2020.

[7] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," 2020.

[8] Y. M. Asano, C. Rupprecht, and A. Vedaldi, "Self-labelling via simultaneous clustering and representation learning," 2020.

[9] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," 2021.

[10] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent: A new approach to self-supervised learning," 2020.

[11] P. Engstler, L. Melas-Kyriazi, C. Rupprecht, and I. Laina, "Understanding self-supervised features for learning unsupervised instance segmentation," 2023.

[12] O. Siméoni, G. Puy, H. V. Vo, S. Roburin, S. Gidaris, A. Bursuc, P. Pérez, R. Marlet, and J. Ponce, "Localizing objects with self-supervised transformers and no labels," 2021.

[13] X. Wang, Z. Yu, S. D. Mello, J. Kautz, A. Anandkumar, C. Shen, and J. M. Alvarez, "Freesolo: Learning to segment objects without annotations," 2022.

[14] T. Ishtiak, Q. En, and Y. Guo, "Exemplar-freesolo: Enhancing unsupervised instance segmentation with exemplars," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15424–15433, June 2023.

[15] W. V. Gansbeke, S. Vandenhende, and L. V. Gool, "Discovering object masks with transformers for unsupervised semantic segmentation," 2022.

[16] J.-J. Hwang, S. X. Yu, J. Shi, M. D. Collins, T.-J. Yang, X. Zhang, and L.-C. Chen, "Segsort: Segmentation by discriminative sorting of segments," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7334–7344, 2019.

[17] L. Melas-Kyriazi, C. Rupprecht, I. Laina, and A. Vedaldi, "Deep spectral methods: A surprisingly strong baseline for unsupervised semantic segmentation and localization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8364–8375, 2022.

[18] J. H. Cho, U. Mall, K. Bala, and B. Hariharan, "Picie: Unsupervised semantic segmentation using invariance and equivariance in clustering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16794–16804, 2021.

[19] M. Hamilton, Z. Zhang, B. Hariharan, N. Snavely, and W. T. Freeman, "Unsupervised semantic segmentation by distilling feature correspondences," *arXiv preprint arXiv:2203.08414*, 2022.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[23] P. Gage, "A new algorithm for data compression," *C Users Journal*, vol. 12, no. 2, pp. 23–38, 1994.

[24] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[25] L. Weng, "The transformer family," *lilianweng.github.io*, Apr 2020.

[26] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[27] S. Qiao, H. Wang, C. Liu, W. Shen, and A. Yuille, "Micro-batch training with batch-channel normalization and weight standardization," *arXiv preprint arXiv:1903.10520*, 2019.

[28] A. Baevski and M. Auli, "Adaptive input representations for neural language modeling," *arXiv preprint arXiv:1809.10853*, 2018.

[29] J. M. Buhmann, J. Malik, and P. Perona, "Image recognition: Visual grouping, recognition, and learning," *Proceedings of the National Academy of Sciences*, vol. 96, no. 25, pp. 14203–14204, 1999.

[30] S. C. Zhu and D. Mumford, "Prior learning and gibbs reaction-diffusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 11, pp. 1236–1250, 1997.

[31] E. Loper and S. Bird, "Nltk: The natural language toolkit," *arXiv preprint cs/0205028*, 2002.

[32] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, pp. 8748–8763, PMLR, 2021.

[33] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[34] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," *arXiv preprint arXiv:1611.00712*, 2016.

[35] A. Van Den Oord, O. Vinyals, *et al.*, "Neural discrete representation learning," *Advances in neural information processing systems*, vol. 30, 2017.

[36] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, *et al.*, "Mlp-mixer: An all-mlp architecture for vision," *Advances in neural information processing systems*, vol. 34, pp. 24261–24272, 2021.

[37] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. Le, Y.-H. Sung, Z. Li, and T. Duerig, "Scaling up visual and vision-language representation learning with noisy text supervision," in *International conference on machine learning*, pp. 4904–4916, PMLR, 2021.

[38] A. Palepu and A. L. Beam, "Tier: Text-image entropy regularization for clip-style models," *arXiv preprint arXiv:2212.06710*, 2022.

[39] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick, "Microsoft coco captions: Data collection and evaluation server," *arXiv preprint arXiv:1504.00325*, 2015.

[40] M. Everingham, L. Van Gool, and W. KI, "C., winn, j., & zisserman, a.(2010)," *The PASCAL Visual Object Classes (VOC) Challenge*, pp. 303–338, 2010.

[41] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 891–898, 2014.

[42] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 633–641, 2017.

[43] S. Changpinyo, P. Sharma, N. Ding, and R. Soricut, "Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3558–3568, 2021.

[44] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li, "Yfcc100m: The new data in multimedia research," *Communications of the ACM*, vol. 59, no. 2, pp. 64–73, 2016.

[45] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[46] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[47] J. Xu, J. Hou, Y. Zhang, R. Feng, Y. Wang, Y. Qiao, and W. Xie, "Learning open-vocabulary semantic segmentation models from natural language supervision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2935–2944, 2023.

[48] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[49] J. Xu, S. De Mello, S. Liu, W. Byeon, T. Breuel, J. Kautz, and X. Wang, "Groupvit: Semantic segmentation emerges from text supervision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18134–18144, 2022.

[50] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.