Perceptual Group Tokenizer: Building Perception with Iterative Grouping

Anonymous Author(s)
Affiliation
Address
email

Abstract

Human visual recognition system shows astonishing capability of compressing visual information into a set of tokens containing rich representations without label supervision. One critical driving principle behind it is perceptual grouping [1, 2, 3]. Despite being widely used in computer vision in the early 2010s, it remains a mystery whether perceptual grouping can be leveraged to derive a neural visual recognition backbone that generates as powerful representations. In this paper, we propose *the Perceptual Group Tokenizer*, a model that entirely relies on grouping operations to extract visual features and perform self-supervised representation learning, where a series of grouping operations are used to iteratively hypothesize the context for pixels or superpixels to refine feature representations. We show that the proposed model can achieve competitive performance compared to state-of-the-art vision architectures, and inherits desirable properties including *adaptive computation without re-training*, and interpretability. Specifically, Perceptual Group Tokenizer achieves 79.7% on ImageNet-1K *self-supervised learning* benchmark with linear probe, marking a new progress under this paradigm.

1 Introduction

Visual recognition mechanisms matter. The pursuit of advanced vision algorithms that encode an image to meaningful representations dates back to late 80s, with two paradigms marking the progress over the past 40 years: feature detection [4, 5, 6, 7] and perceptual grouping [8, 9, 10], where feature detection focuses on specific distinctive patterns, while perceptual grouping considers similarities among all pixels to produce a compact set of tokens as proxies for image representation. Ever since the surge of deep learning, feature detection has predominated the vision field and become the main rationale in representation learning backbone designs and made impressive strides [11, 12, 6, 13, 14, 15, 7]. The success of the former paradigm is, although striking, raising the question of whether perceptual grouping can also be used as the driving principle to construct a visual recognition model.

Different from detecting and selecting distinctive features, perceptual grouping emphasizes on learning feature space where similarity of all pixels can be effectively measured [9, 10]. With such a feature space, semantically meaningful objects and regions can be easily discovered with a simple grouping algorithm and used as a compact set to represent an image [9, 10, 16]. This indicates that image understanding is essentially "pixel space tokenization", and being able to produce generalizable feature representations is connected to whether correct contextual pixels are binded together [17, 18].

The intriguing properties of perceptual grouping, including natural object discovery, deep connections with information theory and compression [19], and association with biological vision system [3] or cognitive science explanations [1], have led to a strong revive recently under deep learning frameworks [16, 20, 21, 22, 23]. However, these methods are either still focusing on small or toy

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

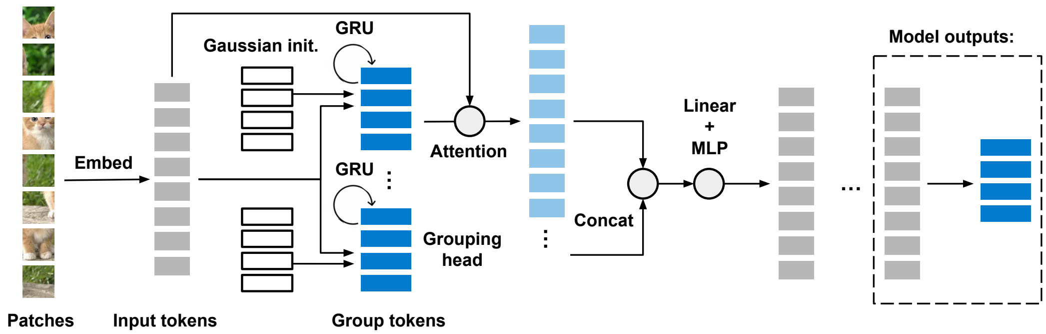

Figure 1: Perceptual Group Tokenizer takes in a sequence of patches (or pixels), generates high-dimensional embedding vectors for all patches, then them passes through a series of grouping layers to refine the embedding vectors as feature representations. Each grouping layer performs K rounds of binding from input tokens to group tokens. To consider various grouping possibilities, multiple grouping heads are adopted. Each group token provides a useful context for input tokens for feature refinement. The final output of the model contains refined input token, group tokens, and assignments between input tokens and groups tokens.

datasets [16, 24, 23], or used as a side add-on component [21] to strengthen the existing vision architectures for increased interpretability. Whether perceptual grouping can be used to build models and learn representations that are as informative and expressive as those learned by state-of-the-art vision architectures remains an open question.

In this paper, we propose *Perceptual Group Tokenizer*, a model trained under *self-supervised learning* framework and building visual representation *entirely relying on perceptual grouping operations*. Given an image, the core of our model is to understand each pixel or patch through hypothesizing its contexts with grouping operations. Starting from given input patches, the grouping operation performs iterative binding process onto a set of randomly sampled group tokens to determine the affinity groups based on similarities. The group tokens are then used as hypothesized contexts to refine the feature representation for the image. We show that applying this simple principle can already produce expressive representations and works well on self-supervised large dataset pretraining.

Compared to self attention, why can grouping operation work? Analyzing the rationale behind it, we build connections from grouping operation to self attention, showing that, if group tokens are treated as communication channels, self attention can potentially automatically emerge during learning processes as a special case, while the grouping operation can produce even richer interactions among tokens. Under this viewpoint, ViT [25] can be considered as a grouping backbone, with a fixed number of grouping slots depending on number of input tokens, and the binding is achieved through stacking more than one layer with non-shared weights. This provides one explanation on why grouping mechanism can be effective on visual representation learning and has the potential to be a promising competitive paradigm for vision architecture designs.

The primary contribution of this work is proposing a new architecture derived purely by perceptual grouping that achieves competitive performance compared to other state-of-the-art architectures on *self-supervised learning* benchmarks, contributing to a new paradigm of developing vision architectures. We thoroughly analyze the design space of perceptual grouping backbones, show the capability of *adaptive computation without re-training*, and visualize the grouping process which produces semantically meaningful bindings among patch tokens.

2 Models

In this section, we introduce Perceptual Group Tokenizer (PGT), a visual recognition architecture entirely driven by perceptual grouping principles. We discuss the core operations for grouping in section 2.1 and the building blocks network architectures in section 2.2 in the main paper, and self-supervision loss and more discussion in section 4.1.2 and 4.1.3 in the supplementary material.

2.1 Perceptual grouping

We start with introducing notations for our method. Given an image $\boldsymbol{x} \in \mathbb{R}^{H \times W \times C}$, we first reshape it as a sequence of small patches[1]. Each patch $\boldsymbol{x}_p \in \mathbb{R}^{h \times w \times c}$ has spatial shape $h \times w$, where $h \ll H$ and $w \ll W$, leading to $N = \frac{HW}{hw}$ number of patches per image. To represent a patch, we embed it into a high-dimensional vector $\boldsymbol{h} \in \mathbb{R}^d$. The set of embedded tokens $\{\boldsymbol{h}_i\}^N$ is referred as *input tokens* in later parts, and used as inputs for the following grouping blocks.

Feature refinement through hypothesizing contexts. One pixel does not have meanings without putting it into contexts. At a high level, image understanding or feature learning is equivalent to binding the correct contextual pixels at all locations. The core idea of our model is to generate many (e.g. over-complete w.r.t number of objects in the image) hypothesized contexts and use the hypothesized contexts as cues to refine the feature representation of each patch. This process is achieved through a grouping module. Given input tokens $\{\boldsymbol{h}_i\}^N$, the grouping module starts from a set of random samples (referred as *group tokens*) from a Gaussian distribution, then performs binding process to aggregate information from input tokens to the group tokens, and ends up with a set of group tokens $\boldsymbol{c}^* = \{\boldsymbol{c}_j^*\}_{j=1}^M$ representing hypothesized contexts among input tokens. The relation between $\boldsymbol{h}_i$ and $\boldsymbol{c}_j$ is soft assigment, indicating how likely an input token belongs to that context. Note that there are often *various ways of generating groupings for an image*, e.g. different semantics, colors, textures, etc., we propose the "multi-grouping operation" to hypothesize rich contexts for tokens. The overall model is shown in figure 1.

Multi-grouping operation. The building block of our model is the multi-grouping operation $\mathcal{G}$, which contains multiple heads to perform the binding process in parallel. This design encourages the model to consider multiple ways of generating groups under different projection spaces. Each head owns a separate Gaussian distribution with learnable means and variance, similar to [26, 16]. Starting from a set of randomly sampled initial group tokens $\boldsymbol{c}_{\text{HEAD}}^{(0)} \sim p(\boldsymbol{\mu}_{\text{HEAD}}, \boldsymbol{\sigma}_{\text{HEAD}})$, the grouping operation uses doubly normalized attention weights to aggregate information from $\boldsymbol{h}$, and the produced group tokens $\boldsymbol{c}_{\text{HEAD}}^{(1)}$ are used for the next round binding. The attention normalization and feature projection are performed in all heads separately.

$$\begin{aligned} \boldsymbol{c}_{\text{HEAD}}^{(1)} &= \mathcal{G}(\boldsymbol{c}_{\text{HEAD}}^{(0)}, \boldsymbol{h}; \theta) \qquad (1) \\ &\cdots \\ \boldsymbol{c}_{\text{HEAD}}^{*} = \boldsymbol{c}_{\text{HEAD}}^{(K)} &= \mathcal{G}(\boldsymbol{c}_{\text{HEAD}}^{(K-1)}, \boldsymbol{h}; \theta) \qquad (2) \end{aligned}$$

where after K steps the final group tokens $\boldsymbol{c}^* = \boldsymbol{c}^{(K)}$ is obtained, and θ is learnable parameters in $\mathcal{G}$. The grouping operator is summarized in algorithm 1.

Implicit differentiation. The iterative grouping process unrolls K steps per operation and leads to heavy burden in the training computation graph. Instead of explicitly backpropagating through the unrolled graph, we follow [24] and treat the multi-grouping process as a fixed point iteration per head. The gradient in the backpropagation is approximated using first-order Neumann series.

2.2 Network architecture

Similar to standard ViT, our model refines the hidden representation $\boldsymbol{h}$ using L model layers. We use $\boldsymbol{h}^l$ to denote the representation after each layer, and explain the design in this section.

Grouping layer. Each grouping layer takes in $\boldsymbol{h}^{l-1}$ as input, and uses the grouping operation in equation 1 to generate group tokens $\boldsymbol{c}_{\text{HEAD}}^* = \{\boldsymbol{c}_{j,\text{HEAD}}^*\}_{j=1}^M$. To use the group tokens to provide context for each $\boldsymbol{h}_i^{l-1}$, we perform another attention operation to obtain the attention matrix (only normalized over group token axis) $\boldsymbol{A} \in \mathbb{R}^{N \times M}$ representing the assignment from input tokens to group tokens, and aggregate the feature back to the input token space:

$$\begin{aligned} \boldsymbol{h}_{\text{HEAD}}^l &= \boldsymbol{A}[\boldsymbol{c}_{1,\text{HEAD}}^*; \boldsymbol{c}_{2,\text{HEAD}}^*; ...; \boldsymbol{c}_{M,\text{HEAD}}^*] \qquad (3) \\ \boldsymbol{h}^l &= \text{Linear}([\boldsymbol{h}_{\text{HEAD}_1}^l; ...\boldsymbol{h}_{\text{HEAD}_H}^l]) \qquad (4) \\ \boldsymbol{h}^l &= \boldsymbol{h}^{l-1} + \text{MLP}(\boldsymbol{h}^l) \qquad (5) \end{aligned}$$

[1]We use 4×4 patches as inputs in this work. Note that our method is generalizable to either pure pixels or other forms of superpixels given a proper patch-to-vector embedding layer.

Method	Arch	Param.	Linear probe (top-1 acc)
SCLR [31]	RN50W4	375	76.8
SwAV [32]	RN50W2	93	77.3
BYOL [32]	RN50W2	93	77.4
DINO [29]	ViT-B/16	85	78.2
SwAV [32]	RN50W5	586	78.5
BYOL [32]	RN50W4	375	78.6
iBOT [33]	ViT-B/16	85	79.5
BYOL [32]	RN200W2	250	79.6
SCLRv2 [34]	RN152w3+SK	794	79.8
DINO [29]	ViT-B/8	85	**80.1**
BEiTv2 [35]	ViT-B/16	85	**80.1**
Ours (PGT-B-256)	PGT-B	70	79.3
Ours (PGT-B-384)	PGT-B	70	79.4
Ours (PGT-B-512)	PGT-B	70	79.6
Ours (PGT-B-768)	PGT-B	70	**79.7**

Table 1: Comparison with strong baselines on ImageNet-1K under linear probe evaluation protocal. PGT-X represents X number of group tokens per grouping layer in inference (same trained model with 256 tokens is used). Our model achieves 79.7%, competitive with state-of-the-art vision backbones, and outperforms ResNet architecrures.

This layer definition follows the standard ViT layer as close as possible, where features from each head are aggregated through concatenation and a linear layer transformation. Each token $\boldsymbol{h}$ is further refined using a follow up multi-layer perceptron.

Grouping blocks. Similar to previous architecture designs [6, 27]. we define blocks for the model. One block contains multiple grouping layers that share the same hyperparameters setups, i.e. number of group tokens, group token dimensions. The full model contains three grouping blocks. This increases the flexibility when exploring model design spaces.

See more detials in sections 4.1.2 and 4.1.3 in the supplementary material.

3 Experiments

We evaluate the representation learned by our model on standard benchmarks, specifically ImageNet-1K dataset. Summarized in the main paper in section 3.1. In the supplementary material, we also thoroughly explore and analyze the design space of perceptual group tokenizer in section 4.2.2, show the adaptive computation ability in section 4.2.3, demonstrate the generalization ability on semantic segmentation in section 4.2.4, and visualize the learned attention in section 4.2.5.

3.1 Main results

Setup. The widely-adopted standard benchmark for evaluating self-supervised learning methods is ImageNet ILSVRC-2012 (ImageNet-1K) [28]. Performance of models are measured by top-1 classification accuracy. The pre-trained backbones are frozen, with a linear classifier trained on top. For fair comparison, we follow the standard data augmentation used in [29], with the same number of global views and local views. The model is optimized using AdamW [30] with learning rate 0.0005 and 1024 batch size for 600 epochs, trained with TPUv5 for 21k core hrs (512 cores for 41 hrs). We use 4×4 patches as image tokens, which keeps as much details as possible while maintaining reasonable computation costs. For machines, we use TPUv5 to run experiments.

The main results are summarized in table 3.1. We mainly compare with ResNet and ViT backbones, the two main stream vision architectures to show that perceptual grouping architecture can also achieve competitive results on the challenging ImageNet-1K benchmark. Although our model is trained with 256 group tokens, the model can use different numbers of group tokens in inference (more experiments in 4.2.2). We evalaute PGT with 256, 384, 512, and 768 number of group tokens and observe that with PGT-768 the model can achieve 79.7% top-1 accuracy, showing the self-supervised learned feature of PGT is as good as the ViT architecture.

References

[1] Stephen E Palmer. Perceptual grouping: It's later than you think. *Current Directions in Psychological Science*, 11(3):101–106, 2002.

[2] Johan Wagemans, James H Elder, Michael Kubovy, Stephen E Palmer, Mary A Peterson, Manish Singh, and Rüdiger Von der Heydt. A century of gestalt psychology in visual perception: I. perceptual grouping and figure–ground organization. *Psychological bulletin*, 138(6):1172, 2012.

[3] Michael H Herzog. Perceptual grouping. *Current Biology*, 28(12):R687–R688, 2018.

[4] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[5] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.

[8] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[9] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104:154–171, 2013.

[10] Pablo Arbeláez, Jordi Pont-Tuset, Jonathan T Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 328–335, 2014.

[11] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[12] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[13] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[14] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[15] Haozhi Qi, Chong You, Xiaolong Wang, Yi Ma, and Jitendra Malik. Deep isometric learning for visual recognition. In *International conference on machine learning*, pages 7824–7835. PMLR, 2020.

[16] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 33:11525–11538, 2020.

[17] Geoffrey Hinton. How to represent part-whole hierarchies in a neural network. *Neural Computation*, pages 1–40, 2022.

[18] Laura Culp, Sara Sabour, and Geoffrey E Hinton. Testing glom's ability to infer wholes from ambiguous parts. *arXiv preprint arXiv:2211.16564*, 2022.

[19] Yi Ma, Harm Derksen, Wei Hong, and John Wright. Segmentation of multivariate mixed data via lossy data coding and compression. *IEEE transactions on pattern analysis and machine intelligence*, 29(9):1546–1562, 2007.

[20] Gamaleldin Elsayed, Aravindh Mahendran, Sjoerd van Steenkiste, Klaus Greff, Michael C Mozer, and Thomas Kipf. Savi++: Towards end-to-end object-centric learning from real-world videos. *Advances in Neural Information Processing Systems*, 35:28940–28954, 2022.

[21] Jiarui Xu, Shalini De Mello, Sifei Liu, Wonmin Byeon, Thomas Breuel, Jan Kautz, and Xiaolong Wang. Groupvit: Semantic segmentation emerges from text supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18134–18144, 2022.

[22] Ziyi Wu, Nikita Dvornik, Klaus Greff, Thomas Kipf, and Animesh Garg. Slotformer: Unsupervised visual dynamics simulation with object-centric models. In *The Eleventh International Conference on Learning Representations*, 2022.

[23] Ondrej Biza, Sjoerd van Steenkiste, Mehdi SM Sajjadi, Gamaleldin F Elsayed, Aravindh Mahendran, and Thomas Kipf. Invariant slot attention: Object discovery with slot-centric reference frames. *arXiv preprint arXiv:2302.04973*, 2023.

[24] Michael Chang, Tom Griffiths, and Sergey Levine. Object representations as fixed points: Training iterative refinement algorithms with implicit differentiation. *Advances in Neural Information Processing Systems*, 35:32694–32708, 2022.

[25] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[26] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[27] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[28] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

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

[30] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. 2018.

[31] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[32] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924, 2020.

[33] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. *arXiv preprint arXiv:2111.07832*, 2021.

[34] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255, 2020.

[35] Zhiliang Peng, Li Dong, Hangbo Bao, Qixiang Ye, and Furu Wei. Beit v2: Masked image modeling with vector-quantized visual tokenizers. *arXiv preprint arXiv:2208.06366*, 2022.

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

[37] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6881–6890, 2021.

[38] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

4 Supplementary Material

4.1 More about models

4.1.1 Algorithm

We provide the pseudo code for the perceptual grouping algorithm as below.

Algorithm 1 Multi-grouping operation.

```
def multi_grouping(h_key, h_value, steps, mu, sigma, num_tokens, num_heads):
  """ Input tensors:
        h_key and h_value are projected multi-head tensors with shape [num_heads x N x d].
  """
  # Initial M group tokens.
  group_tokens = Normal(mean=mu, std=sigma, nsamples=num_tokens)
  group_tokens = group_tokens.reshape(num_heads, num_tokens, d) #[num_heads x M x d]

  # Binding process.
  for step in range(steps):
    # Implicit differentiation.
    if step == steps - 1:
      group_tokens = stop_gradient(group_tokens)
    # Attention operation for group assignment.
    attn_matrix = attention(group_tokens, h_key) #[num_heads x N x M]
    attn_matrix /= attn_matrix.sum(-2, keep_dim=True)
    h_updates = einsum("hij,hid->hjd", attn_matrix, h_value) #[num_heads x M x d]
    group_tokens = gru_cell(h_updates, group_tokens)
    # Grouped mlp/layernorm performs independent mlp/layernorm for each head.
    group_tokens = grouped_mlp(grouped_layer_norm(group_tokens)) + group_tokens

  return group_tokens
```

4.1.2 Self-supervision loss

Following the student-teacher self-supervision loss [29, 36], we use a moving average of online network (student model) as the teacher model to perform representation learning. To summarize group tokens outputed from the final layer, we use one multi-head attention layer with a learnable token to attend to all group tokens. The produced single vector is treated as the feature representation for the image and is input to the loss function.

4.1.3 Discussion

Our proposed model, perceptual group tokenizer, is free of self attention operation and relies purely on grouping operations. In this section, we link the grouping process to several techniques and discuss the rationale on why the model can be effective on representation learning.

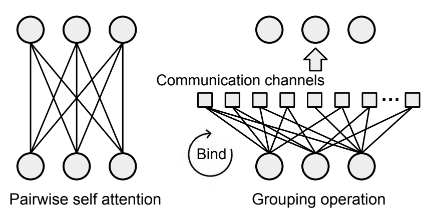

Figure 2: Operation comparison.

Group tokens as "communication channels". The core of feature representation learning is how information is exchanged among pixels. In perceptual grouping backbones, we can consider the set of group tokens as communication channels, where information from different input tokens are aggregated in various ways. Each group token represents a high-order channel that links input tokens with high affinity under certain projected space to exchange information among them. As a thought experiment, if each input token is solely assigned to a different group token (given enough group tokens), then the perceptual grouping layer is equivalent to one self attention layer (up to some engineering design difference). While self attention layers mainly rely on pairwise communications, grouping operation, hypothetically, can automatically learn and emerge both pairwise and higher-order information exchange through the group token communication channels. This can also be linked to traditional *factor graphs* in probabilistic graphical models. Through the lens of that, grouping is forming factor nodes automatically through the learning processes. Under properly designed loss and grouping operation, it has the potential to be more effective if adopting a per-layer comparison between self attention and grouping operation.

Efficiency. Due to the flexibility in customizing number of group tokens (controlled by initial number of samples), grouping operation does not require a strict $O(N^2)$ operation and is $O(NM)$

	Descend	Flat	Ascend
Token size	[576, 384, 192]	[384, 384, 384]	[192, 384, 576]
Accuracy	62.0	63.1	**63.4**
Token shape	[192, 128, 64]	[128, 128, 128]	[64, 128, 192]
Accuracy	63.6	**63.7**	63.1

Table 2: Exploring the design choices for PGT. Token size: dimensions for group tokens in three grouping blocks. Token shape: number of tokens for group tokens in three grouping blocks. Accuracy measured on ImageNet-1K under linear probe protocal. Results indicate progressively large group token dimensions with flat or descend number of tokens arrangements work the best.

on complexity. Furthermore, we show that *even in inference time*, number of group tokens can be adaptively customized, given an already trained model.

4.2 More about experiments

4.2.1 Main results

Architecture details. In the experiments, we mainly evaluate two variants of PGT: the main model and a tiny version for exploring design choices. On the ImageNet-1K benchmark, we report the numbers of our main model. Three grouping blocks are used, with 10 grouping layers in each block. The dimension for input token is 384, with 256 group tokens per layer. The dimensions for group tokens are 98, 192, and 288 for the three blocks, respectively. There are 6 grouping heads used. For number of grouping iterations, we observe three rounds are enough. The MLP hidden size for each layer is 384 as well, i.e. MLP multiplication factor is 1. The final multihead attention layer uses a learnable token with 2048 dimensions to summarize all group tokens outputs from the model.

4.2.2 Ablations

To explore design choices of PGT, we adopt a tiny version with 3 blocks, 2 layer in each block (6 layers in total), 256 hidden size for input tokens, and 3 number of grouping iterations. The learnable token in MAP head has 512 dimensions. There are around 10M parameters in a PGT-tiny model.

Group token layouts. Given a fixed number of budget on group tokens, we explore three choices on how they should be arranged across grouping blocks and layers: descend, flat and ascend. Intuitively, more group tokens will have higher capacity of capturing smaller parts and detailed visual features, while less group tokens are more prone to carry global information. As shown in table 4.2.2 bottom row, flat or descend number of group tokens performs the best. In practice, we find that using flat (same number of group tokens in three grouping blocks) version has better stability in training.

Group token dimension shapes. Similar to token number arrangements, we explore how group token dimensions should be set. Under three choices, progressively increasing the dimension size in the later layers performs the best, shown in first row of table 4.2.2. This also aligns with the intuition that later layers contain more information and requires higher capacity to represent groups.

Multi-grouping vs single grouping. We further tests whether multi-head grouping helps improve performance. As a fair comparison, we use 6 heads and 128 group tokens per head for multi-grouping model, and 1 head with 6×128 group tokens for the single grouping model. We find that adopting multi-head design can improve the performance from 62.2% to 66.3%, a 4.1% accuracy boosts, showing that having multiple heads indeed helps with representation learning.

Grouping distribution entropy. Will grouping process collapse to some specific group token during training? We visualize the entropy of marginal distribution over tokens $p(\boldsymbol{c})$ and conditional distribution $p(\boldsymbol{c}|\boldsymbol{x})$ in figure 3 and 4. Interestingly, we observe that conditional probability, i.e. the assignment to group tokens, tends to become more certain during training, while the marginal distribution remains having descend entropy. This indicates that collapse does not happen in the self-supervised training process.

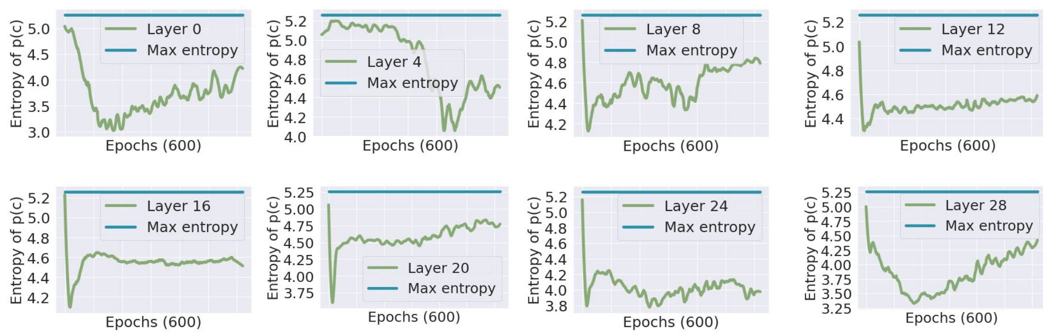

Figure 3: The entropy curves of marginal distribution $p(\boldsymbol{c})$ grouping across different layers.

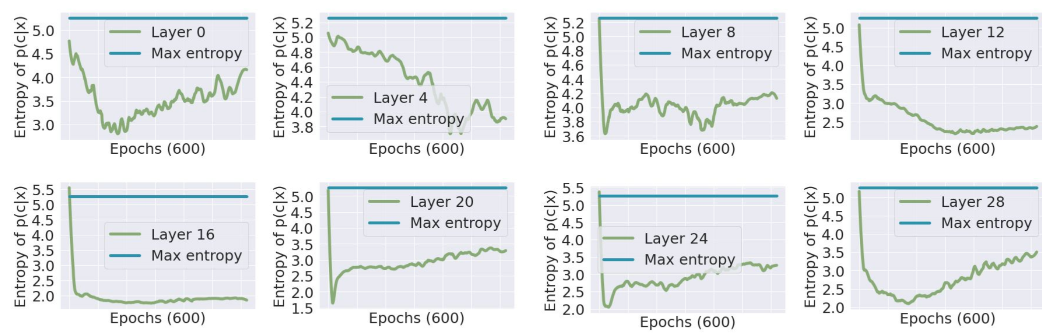

Figure 4: The entropy curves of conditional distribution $p(\boldsymbol{c}|\boldsymbol{x})$ grouping across different layers.

4.2.3 Out-of-distribution adaptive computation

One surprising and powerful ability of PGT is adaptive computation. Because the initial group tokens are sampled from a Gaussian distribution, the number of group tokens can be flexibly customized in inference time given a trained model. This leads to an out-of-distribution adaptive computation ability customizable according to needs, e.g. computation or tasks. We mainly test PGT-Tiny with a grid evaluation of different number of group tokens used in training and inference. PGT-B model with 256 group tokens is also tested under different inference token budgets. Results are summarized in table 4.2.3. Our model shows strong out-of-distribution generalizability, indicated by the results. Surprisingly, *with more number of tokens, the performance can be increased*. When using the larger main model PGT-B to perform adaptive inference, with only 12.5% of the number of group tokens compared to training, the performance can still be maintained at 71.18% with only a 8% drop on top-1 accuracy.

4.2.4 Semantic segmentation on ADE20k

To evaluate the generalizability of pretrained feature produced by PGT, we test the transfer performance of semantic segmentation with ADE20k. Following the standard setup, we finetune our model with the same data augmentation for 128 epoch. The baseline method uses DINO + ViT-B/16, and is fine-tuned with a SETR-PUP segmentation head [37]. For our model, we only add one linear classification layer after the pre-trained PGT for fine-tuning. To adapt to more objects and complex scenes in the segmentation datasets, we use 1024 group tokens for inference, benefiting from the adaptive computation ability of our model. We find that our model (PGT-B + Linear) can obtain 44.5% on mean IoU while the baseline (ViT-B/16 + SETR-PUP) achieves 44.1% [38], leading to a 0.4% improvements.

tr/inf	16	32	64	128	256	384
PGT-Ti-16	$\underline{57.44}$ ($\times 1$)	58.28 ($\times 2$)	**58.49** ($\times 4$)	58.47 ($\times 8$)	58.48($\times 16$)	58.39 ($\times 24$)
PGT-Ti-32	57.33 ($\times \frac{1}{2}$)	$\underline{59.86}$ ($\times 1$)	60.82 ($\times 2$)	**61.01** ($\times 4$)	60.99 ($\times 8$)	60.89 ($\times 12$)
PGT-Ti-64	53.02 ($\times \frac{1}{4}$)	59.20 ($\times \frac{1}{2}$)	$\underline{61.68}$ ($\times 1$)	62.55 ($\times 2$)	62.91 ($\times 4$)	**62.92** ($\times 6$)
PGT-Ti-128	44.86 ($\times \frac{1}{8}$)	56.63 ($\times \frac{1}{4}$)	61.80 ($\times \frac{1}{2}$)	$\underline{63.88}$ ($\times 1$)	64.66 ($\times 2$)	**64.82** ($\times 3$)
PGT-Ti-256	27.20 ($\times \frac{1}{16}$)	47.35 ($\times \frac{1}{8}$)	58.80 ($\times \frac{1}{4}$)	63.32 ($\times \frac{1}{2}$)	$\underline{65.13}$ ($\times 1$)	**65.52** ($\times \frac{3}{2}$)
PGT-Ti-384	26.09 ($\times \frac{1}{24}$)	43.02 ($\times \frac{1}{12}$)	55.35 ($\times \frac{1}{6}$)	61.67 ($\times \frac{1}{3}$)	64.57 ($\times \frac{2}{3}$)	$\underline{\mathbf{65.50}}$ ($\times 1$)
PGT-B-256	58.18 ($\times \frac{1}{16}$)	71.18 ($\times \frac{1}{8}$)	76.51 ($\times \frac{1}{4}$)	78.39 ($\times \frac{1}{2}$)	$\underline{79.16}$ ($\times 1$)	**79.47** ($\times \frac{3}{2}$)

Table 3: Out-of-distribution adaptive computation by selecting different numbers of initially sampled tokens. Row: number of tokens used for training. Column: number of tokens used for inference. Top-1 accuracy is reported under linear evaluation protocol using ImageNet-1K. The reported performance of first six rows is obtained using a tiny version of PGT, and last row is the main model. Number of group tokens is the same for underlined numbers in training and inference. **Bold numbers** are the best results.

4.2.5 Grouping visualization

We further visualize the generated attention maps during grouping processes to inspect the behaviour of grouping operations. In figure 5, the patch to group token attention maps across all grouping iterations are shown. We find that even the first iteration step can sometimes generate meaningful attention maps. With more iterations, attention maps are more focused on meaningful regions. Figure 6 shows attention maps across different layers of PGT-B. We observe that early layers tend to capture fine-grained elements, while the last layer focuses on semantic information. Multiple grouping heads are indeed capturing various ways of grouping image features, for example, in the first image, first group focuses on color and lights, second head relies on spatial cues, while the last one potentially captures textures.

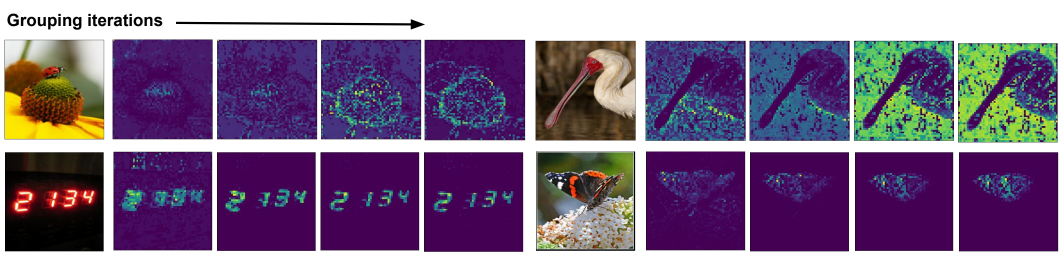

Figure 5: Attention maps produced along grouping processes in the last layer of PGT.

4.3 Conclusion

In this paper, we propose Perceptual Group Tokenizer (PGT), a new visual recognition architecture entirely built through perceptual grouping principles. The proposed model shows strong performance on self-supervised learning benchmark ImageNet-1K with linear probe evaluation, and has desirable properties such as adaptive computation and interpretability in each operation. This work potentially opens a new paradigm for designing visual recognition backbones and hopes to inspire more research progress along this direction. One limitation of the proposed model is its relatively expensive computation cost due to iterative binding processes. This can be potentially addressed by other grouping operations, for example grouping operations with closed-form solutions. We leave this to future works.

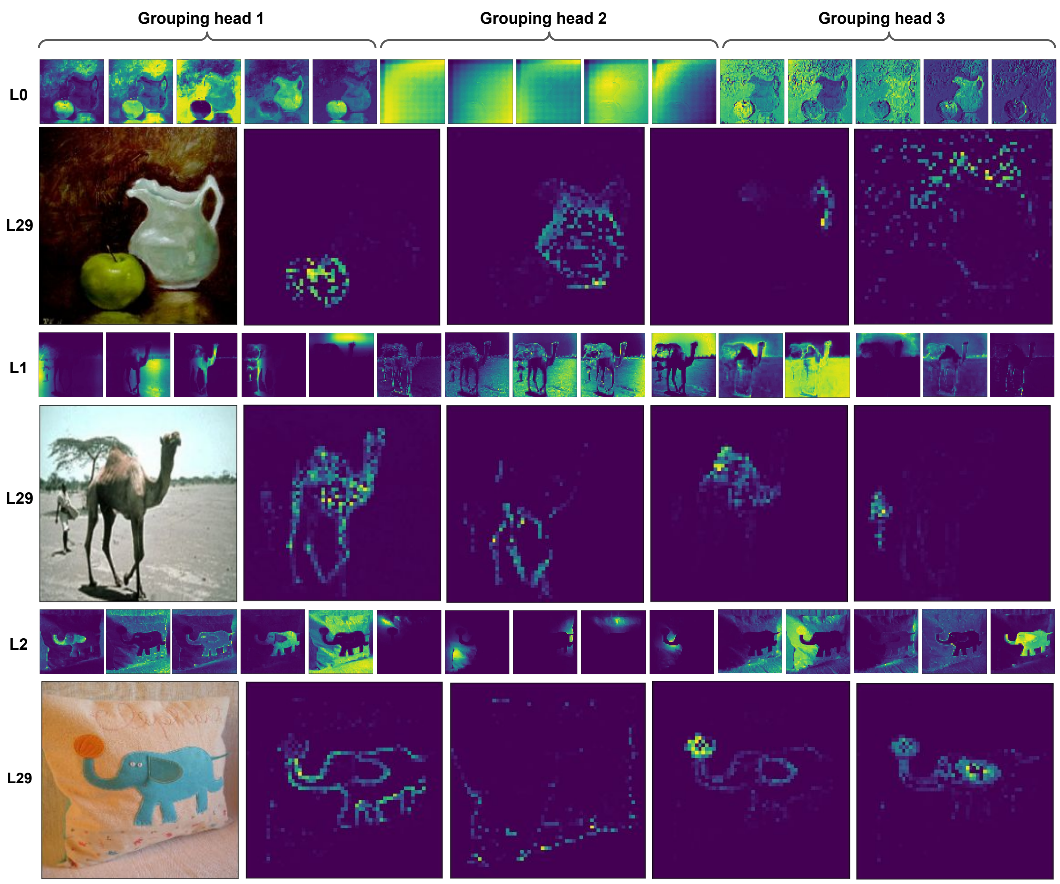

Figure 6: Visualization of attention maps of each group tokens across layers and grouping head. L indicates layer indices. Five group tokens for each grouping head. Smaller images are for early layers, arranged as five group tokens per grouping head. Zoom in for better viewing. Large images are for the last layer.