EFFICIENT TRANSFORMER-BASED PANOPTIC SEGMENTATION VIA KNOWLEDGE DISTILLATION

BY

WENTAO ZHANG

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois Urbana-Champaign, 2023

Urbana, Illinois

Adviser:

Research Assistant Professor Liangyan Gui

ABSTRACT

Knowledge distillation has been applied to various models in different domains. However, knowledge distillation on panoptic segmentation has not been studied so far. In this work, we focus on the knowledge distillation on transformer-based model. More specifically, we perform thorough analysis on the Mask2Former model, which is one of the state-of-the-art models. We found that both backbone and segmentation head are bottleneck of the model performance. To build an efficient transformer-based panoptic segmentation model, one of the best practice is to direct initialize the student model with part of the teacher's parameters. We first worked on layer parameter initialization and parameter group consistent parameter selection for initialization. We then explored different distillation matching schemes between layers and of teacher and student. Finally, we researched different distillation loss, including adaptive matching-based prediction loss, masked generative distillation-based image feature loss, standard attention distillation loss, and deformable attention distillation loss. With all distillation approaches mentioned above, we trained Mask2Former-S(hallow), Mask2Former-T(hin), and Mask2Former-ST. Our ResNet-50 based models outperformed previous strong baselines, including Panoptic Segformer, MaX-DeepLab, MaskFormer, DETR, Panoptic-DeepLab and Panoptic-FPN with far fewer parameters and GFlops on MS COCO dataset. Additionally, our ResNet-18 based model ourperformed ResNet-50 based Panoptic-DeepLab and Panoptic-FPN with only 29.3% of the parameters.

ACKNOWLEDGMENTS

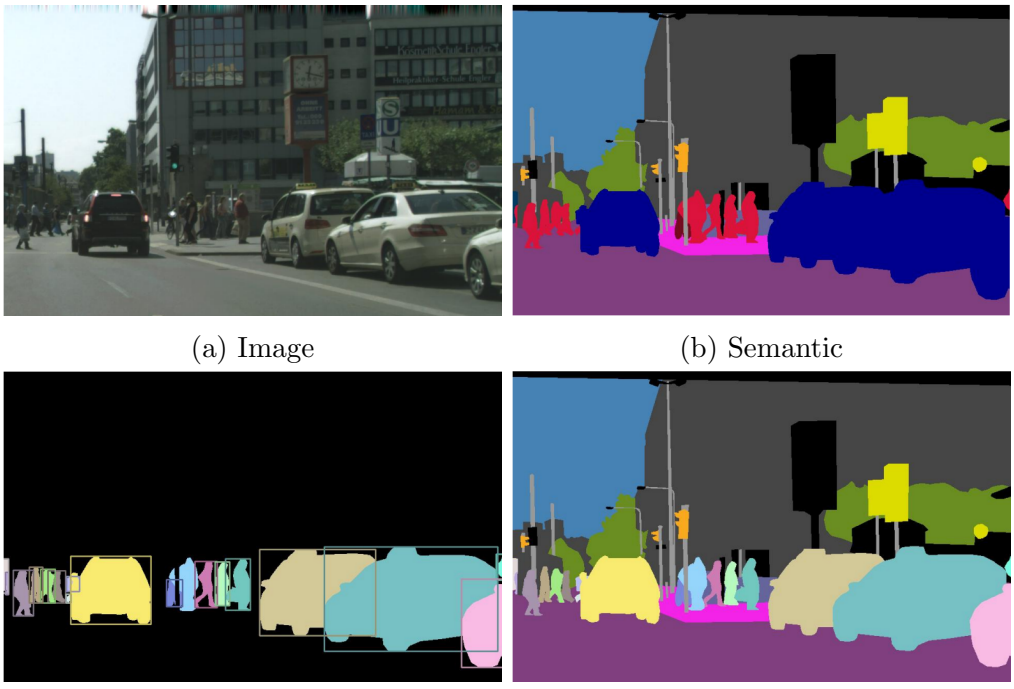
I would like to express my gratitude to Professor Liangyan Gui for her invaluable assistance and foundational insights during my master's program and throughout the process of writing this thesis.

I would also like to extend my thanks to Shengcao Cao, whose timely guidance and ideas greatly contributed to the success of this research project.

Finally, this work used Graphics Processing Unit (GPU) servers at National Center for Supercomputing Applications (NCSA) Delta through allocation CIS220014 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION



(a) Image (b) Semantic

(c) Instance (d) Panoptic

Figure 1.1: An example of different types of image segmentation from [1].

Image segmentation researches mainly focus on grouping pixels into different categories. These categories and memberships lead to different tasks. Two major categories that have been studied by the communities are *stuff* and *things*. *Stuffs* refers to regions of similar texture, such as sky, glass, water, and road. *Things* refer to countable objects like peoples, vehicles, and animals. The task working on grouping pixels by stuff types is usually known as *semantic segmentation*, while studying things is typically formulated as *object detection* and *instance segmentation*. Although these two tasks seem similar, the datasets, metrics and research details of them vary substantially. As a result, a novel task named *panoptic segmentation* that reconcile between instance segmentation and semantic segmentation is proposed recently. Panoptic segmentation unifies things and stuff segmentation and defines a uniform evaluation metrics. Figure 1.1 shows examples of different type of image segmentation.

Numbers of researches are done on panoptic segmentations. [1] first proposed and formulated the panoptic segmentation tasks. A lot of previous works, such as UPSNet [2], DetectorRS [3], and EfficientPS [4] have tried to build panoptic segmentation model based on Feature Pyramid Network [5] and Mask R-CNN [6] style models. These models have achieved competitive performance.

1.1 TRANSFORMER-BASED PANOPTIC SEGMENTATION

Recently, the growing interest in exploiting the potential of Transformer architecture in computer vision has been fueled by the success of large-scale pretrained language models such as GPT-3 [7]. An increasing number of researchers started trying to exploit the potential of Transformer architecture. DEtection TRansformer(DETR) [8] first proposed an end-to-end Transformer-based dense vision prediction framework, which has served as a foundation for subsequent research in this area. Researchers have since developed Transformer-based image segmentation models that have achieved state-of-the-art performance compared with specialized architectures based on the DETR-like architecture.

Specifically, MaX-DeepLab [9] is the first transformer-based end-to-end panoptic segmentation model. MaskFormer [10] decoupled mask prediction and class prediction, and unified different segmentation task in one framework. Mask2Former [11] further improved the MaskFormer via cross attention between image feature and query. Panoptic Segformer [12] first leveraged deformable attention to efficiently leverage multi-scale image features. Mask-DINO [13] extended the DINO [14] via adding a mask prediction branch. These architectures have demonstrated remarkable success in advancing the state-of-the-art (SOTA) performance in panoptic segmentation, as evidenced by the results presented in recent research publications in the field.

1.2 KNOWLEDGE DISTILLATION FOR TRANSFORMER-BASED PANOPTIC SEGMENTATION

However, recent works for universal image segmentation are not fast enough. Mask2Former only has a 9.7 frame rate with ResNet-50 backbone, and MaskedDINO has a 14.8 frame rate on MS COCO 2017 dataset with one NVIDIA A100 GPU. There is still a gap between research work and practical applications in terms of inference speed. Previous works, such as EfficientPS, have tried to build efficient panoptic segmentation models via sophisticated backbone and head design. In this work, we tried to leverage knowledge distillation to build efficient transformer-based panoptic segmentation model.

Knowledge distillation is proposed by [15]. Recent progress on knowledge distillation for dense prediction is mainly focused on object detection [16, 17, 18, 19, 20, 21] and semantic segmentation [22, 23, 24, 25]. There are also works that focus on general feature and logits distillation for all image recognition task, such as [26, 27, 28, 29, 30, 31]. These works mainly focus on convolution based image recognition models. With the rise of Vision Transformers and Pretrained Transformer-based Language Model, researchers also built faster, more

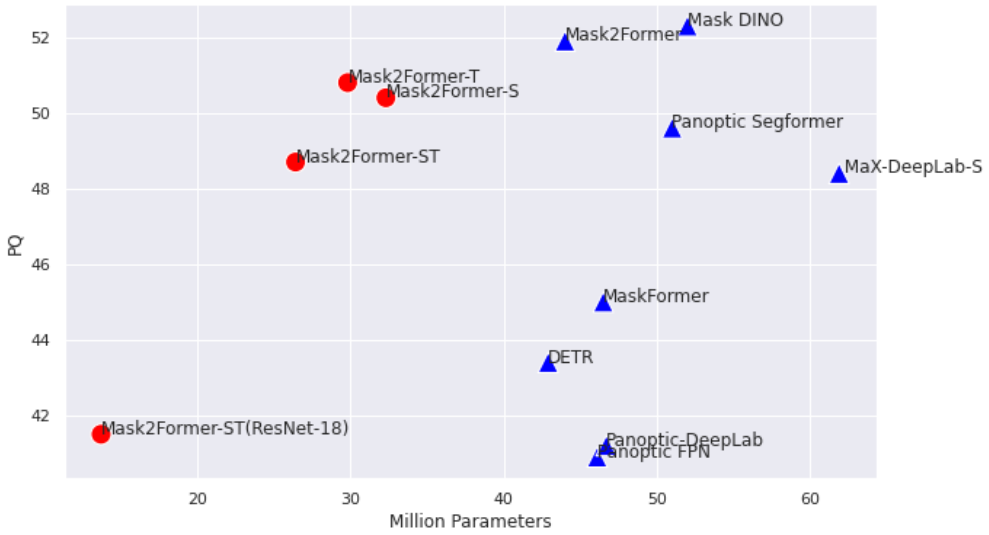


Figure 1.2: The performance of our models (red) compared with existing baselines (blue) with different parameters. The figure shows that our models have achieved great performance that is comparable with current baselines with far less parameters.

lightweight, and more eco-friendly transformer-based models via knowledge distillation.

In prior literature, the study of transformer knowledge distillation has been primarily focused on language model distillation [32, 33, 34, 35] and vision transformers [36, 37, 38, 39, 40]. Moreover, several investigations have delved into knowledge distillation for dense prediction tasks. Chang et al. [41] introduced an effective yet straightforward method distill teacher's knowledge from different components of teacher model. KD-DETR [42] presented an innovative sampling strategy to separate detection from knowledge distillation, achieving notable improvements in detection tasks. D^3ETR [43] proposed a decoder distillation technique to transfer knowledge between the teacher and student through decoder attention maps and predictions. Nonetheless, no existing research has addressed efficient panoptic segmentation.

In this work, we first made analysis on recent state-of-the-art transformer-based panoptic segmentation model Mask2Former, and found that both backbone and segmentation head are bottleneck of the efficiency. The backbone only accounts for 53% of parameters and 32% of computation. To reduce the computation but maintain the performance, we need to use both a smaller backbone and a smaller segmentation head. As a result, we proposed an effective approach to build smaller and more efficient segmentation model from teacher model. The approach first directly take a part of teacher's parameters to initialize the student model using layer selection or parameter-group consistent parameter selection, and then distill the teacher to build a smaller model via matching based prediction distillation, attention

distillation, deformable attention distillation, and image feature distillation. To minimize the performance loss, we carefully explored on parameter reduction in terms of depth and width of the segmentation head as well as the backbone, and proposed a corresponding knowledge distillation scheme. The scheme includes:

1. Student weight initialization using selected teacher's layers.
2. Student weight initialization using sparsity consistent structural pruned teacher.
3. Layer-wise matching strategy for distillation.
4. Adaptive prediction matching for distillation.
5. Self and cross attention distillation based on the matching result.
6. Deformable attention distillation for image feature encoder.

Furthermore, we also found that applyinng this approach in an iterative way can yield smaller but more efficient panoptic segmentation model compared with directly distilling from teacher. We tested our approach based on Mask2Former to build **Mask2Former-T**(hin), **Mask2Former-S**(hallow) and **Mask2Former-ST** with ResNet-50 [44] and ResNet-18 [44] backbone using MS COCO [45] dataset.

The experiment result shows that our efficient Mask2Former-S and Mask2Former-T outperforms various previous baselines such as Panoptic-DeepLab, MaX-DeepLab, DETR, Panoptic Segformer and MaskFormer with less parameters and less computation on MS COCO dataset using ResNet-50 backbone. Our further experiment showed that using our proposed multi-stage distillation, a MaskFormer-ST model with ResNet-18 backbone can outperform previous ResNet-50 based model with only 13.7 million parameters and only 68 Gflops computation given an input of size 1024×1024. Figure 1.2 shows the comparison of our models and existing baselines in terms of number of parameters and performance.

CHAPTER 2: PRELIMINARY

2.1 MODEL ARCHITECTURE ANALYSIS

Before starting knowledge distillation, we need to analyze the model architecture, model components and how each part contributes to the computation of those Transformer-based image segmentation models. Figure 2.1 illustrate the general architecture of Transformer-based panoptic segmentation models. The mask prediction can be decomposed into following steps:

1. The backbone encodes the input image $\mathbf{I} \in \mathbf{R}^{C \times H \times W}$ to obtain multi-scale features. Common image feature backbones such as ResNet [44], ViT [46], and Swin Transformer [47] are frequently used for this purpose. Here, C is the number of input channel, and H, W is the height and width of input image. Usually $C = 3$.

2. The multi-scale features $\mathbf{F} \in \mathbf{R}^{C_{\text{hidden}} \times (H/32 \times W/32 + H/16 \times W/16 + H/8 \times W/8 + H/4 \times W/4)}$ are then fed into an image feature encoder to refine the multi-scale feature obtained from the backbone. For Mask2Former, a transformer with Deformable Attention [48] is used.

3. The multi-scale features are then up-sampled by a mask generator, resulting in a pixel embedding $\mathbf{E}_{\text{pixel}} \in \mathbf{R}^{\mathcal{E} \times H \times W}$. Each generated mask has a correspondence with a specific thing, stuff in the image or "nothing" ($\emptyset$). Here, $\mathcal{E}$ represents the embedding dimension of one mask.

4. To decode the mask and decide the class of each mask, a transformer decoder with trainable query embeddings is used. The transformer decoder accepts both queries $\mathbf{Q} \in \mathbf{R}^{N \times \mathcal{E}}$ and multi-scale features, and finally outputs mask embedding $\mathbf{E}_{\text{mask}} \in \mathbf{R}^{N \times \mathcal{E}}$. Where N is the pre-defined number of quries.

5. A dot product between $\mathbf{E}_{\text{mask}}$ and $\mathbf{E}_{\text{pixel}}$ is performed to generate binary masks $\mathbf{M} \in \mathbf{R}^{N \times H \times W}$ for queries. To decide which class each mask belong to, a classifier is then applied to the mask embedding to get the classification score of each mask $\mathbf{S} \in \mathbf{R}^{N \times (K+1)}$, where K is the number of classes of things and stuffs, and the extra output dimension correspond to "nothing".

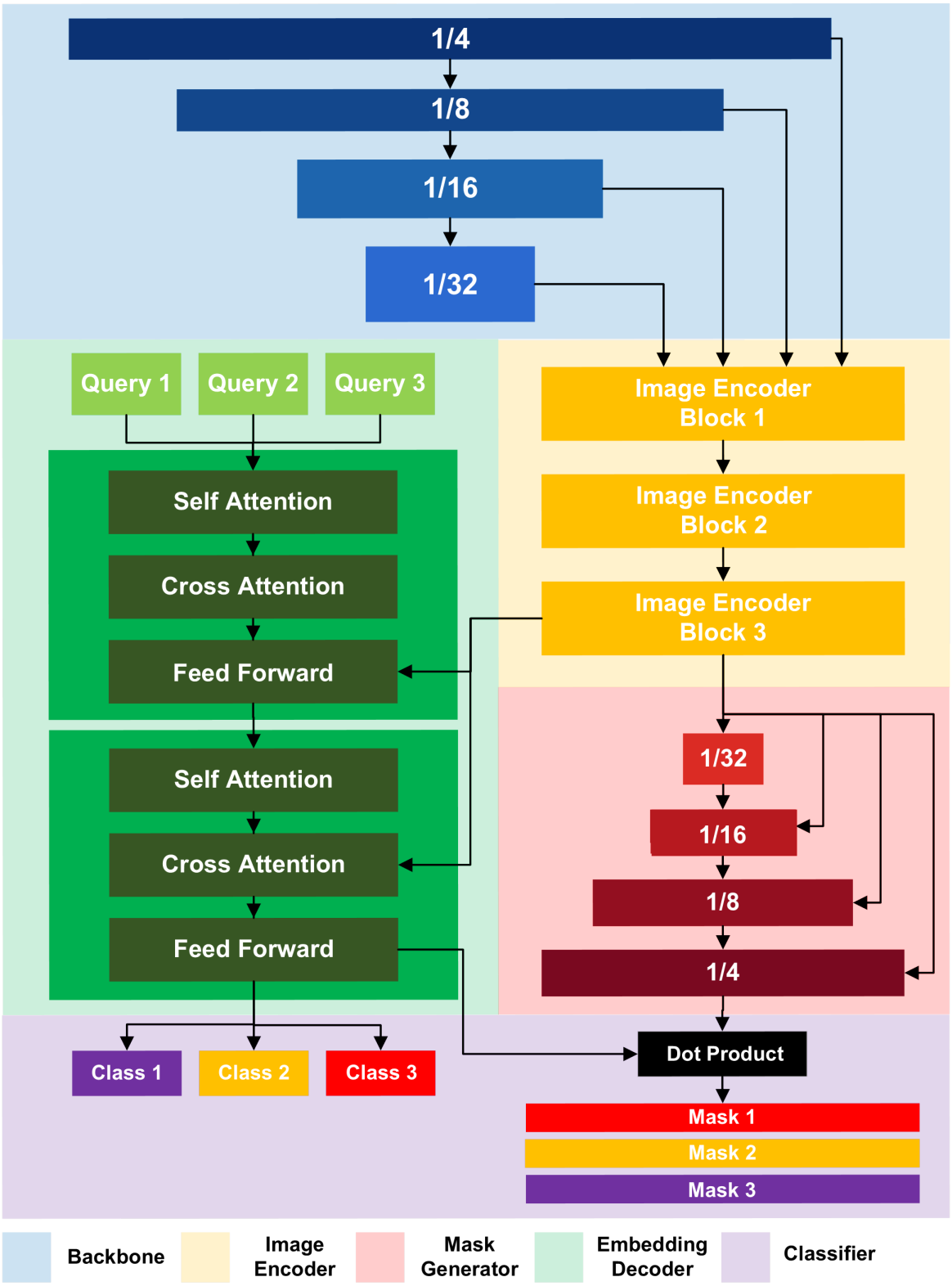


Figure 2.1: General architecture of Mask2Former panoptic segmentation models.

2.2 COMPUTATION AND PARAMETER BREAKDOWN

Besides the model architecture, we also need a insight of parameters and computation breakdown of the Mask2Former model we research on so that we can design the corresponding distillation scheme.

Module Name	#P(Million)	%P	#F(Gflops)	%F
Backbone	23.45	53.3%	71.29	32.1%
Image Encoder	5.31	12.1%	82.82	37.3%
Mask Generator	0.71	1.6%	39.36	17.7%
Embedding Decoder + Classifier	14.49	32.9%	28.60	12.9%
Total	43.99	100%	222.07	100%

Table 2.1: Computation and parameter breakdown of Mask2Former with input size 1024×1024 and ResNet-50 backbone. The result is based on 100 image average on MS COCO dataset. Here, #P means the number of parameters, #F means the floating point operations per second(flops), %P means the percentage of parameters, %F means the percentage of flops.

Table 2.1 presents a comprehensive breakdown of the computational complexity and number of parameters of the Mask2Former model. This breakdown highlights that, in addition to the backbone, several other components also contribute significantly to the overall parameters and computation. While prior distillation techniques have mainly focused on building a model with a smaller backbone, the breakdown emphasizes the importance of distilling from a large head to a smaller one as well.

2.3 COMPLEXITY ANALYSIS

To determine the contributing factors of computation in the Mask2Former model,

we performed a complexity analysis of its major components.Since the backbone network's complexity and design have already been extensively studied and optimized, and we can easily adopt a more efficient backbone network for our model. Thus, we focused on evaluating the complexity of the remaining parts of the model. By conducting this analysis, we were able to identify areas for potential optimization and efficiency gains. Table 2.2 illustrates the meaning of notations used in this section.

2.3.1 Transformer with Deformable Attention

The transformer with deformable attention is used in encoder for refining the multi-scale feature of size $\left(\frac{HW}{r_1} + \frac{HW}{r_2} + \frac{HW}{r_3}\right) \times d = cHW \times d$, since r_1, r_2, r_3 can be considered as

Notation	Meaning of Notation
L_e	Number of layers in image feature refinement encoder
L_d	Number of layers in embedding decoder
d	Embedding size of encoder and decoder
h_e	Intermediate dimension of feed forward network in image refinement encoder
N_s	Number of sampled neighbors in deformable attention
h_d	Intermediate dimension of feed forward network in mask embedding decoder
m_e	Number of heads used for image refinement encoder
m_d	Number of heads used for image mask embedding decoder
H	Height of input image
W	Width of input image
r_i	Resolution factor, $r_i = 2^{i+2}$
N_q	Number of queries
k	Upsample convolution kernel size
C	Number of classes of things and stuff
c	Constant

Table 2.2: Notation used for complexity analysis

constants. Here $\frac{HW}{r_1}$, $\frac{HW}{r_2}$, and $\frac{HW}{r_3}$ corresponds to multi-scale feature outputs of the backbone. For each block of transformer with deformable attention, it calculates the offset for location sampling and then calculates attention scores using linear projection for each head. This yields the complexity: $\underbrace{\mathcal{O}(cHWdm_e \times 2N_s)}_{\text{offset}} + \underbrace{\mathcal{O}(cHWdm_eN_s)}_{\text{attn score}} \sim \mathcal{O}(HWdm_eN_s)$. With the sampled features and attention scores, the complexity of deformable attention can be written as $\underbrace{\mathcal{O}(cHWm_ed^2)}_{\text{sample transform}} + \underbrace{\mathcal{O}(cHWdm_eN_s)}_{\text{aggregation}} + \underbrace{\mathcal{O}(cHWd^2)}_{\text{output transform}}$. Since $m_eN_s << HWd$, we can consider m_e and N_s as constants. So the deformable attention has the complexity $\mathcal{O}(HWd^2)$. The following feed forward network has the complexity $\mathcal{O}(HWdh_e + WHh_ed)$, and usually $h_e = cd$. So the overall complexity of the transformer block is $\mathcal{O}(HWd^2)$, and the entire encoder of complexity $\mathcal{O}(L_eHWd^2)$.

2.3.2 Transformer with Multi-Head Self Attention and Cross Attention

Similar to transformer with deformable attention, the standard transformer has attention module as well as feed forward network. The self-attention part has complexity $\mathcal{O}(N_qd^2 + N_q^2d)$, the cross-attention between query and image feature has complexity $\mathcal{O}(N_qd^2+N_qHWd)$, and the feed forward network has complexity $\mathcal{O}(2N_qdh_d)$. Usually $h_d = cd$, this result in the overall complexity $\mathcal{O}(2N_qd^2 + N_q^2d + N_qHWd)$ of transformer decoder block. And overall complexity $\mathcal{O}(L_d(2N_qd^2 + N_q^2d + N_qHWd))$ of entire transformer decoder since decoder has

L_d layers.

2.3.3 Upsampling Layer

The upsampling is a bilinear interpolation following an output convolution. The upsampling has the complexity $\mathcal{O}(HW/r_i)$, and the convolution has the complexity $\mathcal{O}(k^2HWd^2/r_i)$. While k and r_i are small constants, the overall complexity of upsampling is $\mathcal{O}(HWd^2)$.

2.3.4 Classifier

The classifier compose of two parts, the mask classification and dot product between query embedding and mask embedding. The former can be a linear transformation from embedding space to label distribution. It can be consider as the dot product between two matrix of size (N_d, d) and (d, C). It has the complexity $\mathcal{O}(N_q dC)$. The latter can be consider as the dot product between query vectors and mask embedding. The mask embedding is of shape (WH, d) and the query vectors are of size (N_q, d). So it has complexity $\mathcal{O}(WHdN_q)$.

As a result, the overall complexity of the panoptic segmentation head is:

$$\mathcal{O}\left(L_e HWd^2 + L_d(2N_q d^2 + N_q^2 d + N_q HWd) + HWd^2 + N_q dC + HWdN_q\right) \tag{2.1}$$

In the formula 2.1, W, H, C, and N_q are variables that control negligible number of trainable parameters. So the only parameter we can reduce are the encoder layer L_e, decoder layer L_d, and the embedding size d. To efficiently reduce L_e, L_d and d, we explored different parameter initialization and distillation approaches in the methodology part.

CHAPTER 3: METHODOLOGY

3.1 OVERVIEW

In this part, we will discuss how to efficiently initialize the student model with teacher's parameter, and how to effectively distill the teacher models with smaller L_e, L_d, and d.

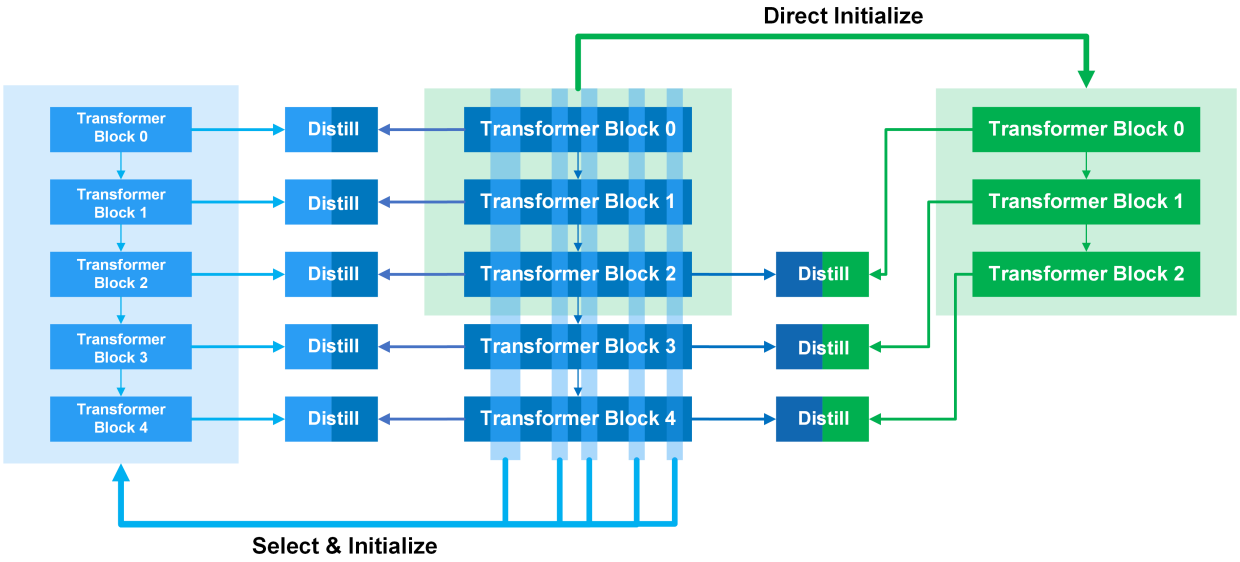


Figure 3.1: Overview of two approaches of reducing parameters and computation. The model in the middle is the teacher model. The left part illustrate the model with smaller embedding size whose layers are initialized by selecting part of parameter in corresponding teacher's layer with mask. The model on the right with less layers are initialized by first several layers of the teacher. After initialization, two students further distill the teacher to improve the performance.

Figure 3.1 illustrate two approaches we will discuss to reduce the complexity of the model. We can directly take several layers from the teacher and initialize the "shallow" student (right) with taken parameters. We can also define a binary mask and select part of the teacher's parameter consistently along layers to initialize the "thin" student model (left). In the following parts, we first discuss how to select layers and parameters to initialize smaller and more efficient model, then describe the distillation using output prediction, image feature, deformable attention, self attention and cross attention module. Finally we will also introduce how to apply our proposed distillation in multi-stage distillation.

3.2 LAYER SELECTION FOR STUDENT INITIALIZATION

The prevalent approach in knowledge distillation for transformer models involves initializing the student model with the teacher's parameters. Nevertheless, determining a good teacher's layer selection strategy remains an open question, particularly when the encoder or decoder in the teacher and student models have differing layer counts. Generally, there are three primary layer-matching strategies: First: initialize the student model with first several layers of teacher's model, Last: initialize the student model with last several layers of teacher's model, and Dilated: initialize the i-th student the $2i$-th layer of the teacher.

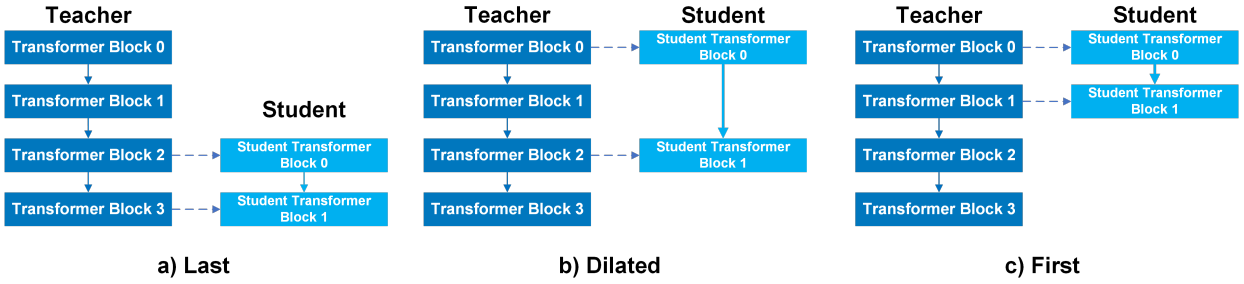


Figure 3.2: Three types of layer weight initialization

In the experiment, we performed ablation study on different initialization approaches and figure out the best strategy among three for both encoder and decoder.

3.3 GROUP-CONSISTENT PARAMETER SELECTION FOR STUDENT INITIALIZATION

To reduce embedding size d, a new panoptic segmentation model with embedding size $d' < d$ but with the same architecture of teacher model is defined. Instead of randomly initialize the new model and then distill from the teacher, there is chance that we can select some parameter consistently from teacher and initialize the student.

Inspired by DepGraph [49], we perform a similar dependent parameter group analysis. Unfortunately, the tools and algorithms provided in DepGraph can hardly apply to the Mask2Former panoptic segmentation scenario and other complicated networks because of vague definition of dependency. As a result, we proposed a more flexible and clearer algorithm based on splitting computation graphs. Given the computational graph $G = (V, E)$, V contains computation operators and tensors, and E decides computation dependency. And the input, which can be a set of tensors $\mathbf{T} = \{T_1, T_2, \cdots, T_n\}$, can be obtained from data such as image and tabular data, or obtained from trainable embedding. Normally, com-

putation graph nodes did not discriminate trainable parameters from other tensors. So we further define Variable and Parameter.

Definition 3.1 (Parameter)**.** A Parameter is a tensor that is trainable in a program.

Definition 3.2 (Variable)**.** A Variable is a tensor in a program that is not a Parameter.

To analyze the dependency of parameters in the computation graph, we followed the idea of "tracking the index of the feature". An extra input: the index of feature is required. Formally, $\mathbf{I} = \{i_1, i_2, \cdots, i_n\}$, where i_j is the index of feature of j-th input tensor T_j. Here "index of feature" indicate the dimension of input represents a "unit" of feature. For instance, the feature dimension of image input of shape (C, H, W) is 0, the dimension of channel. And for text input (L, H) the feature dimension is 1, which is the dimension of embedding of token.

Definition 3.3 (Feature Reduction)**.** Operations in computation graphs that reduce the feature dimension into a scalar is called Feature Reduction.

Some examples of feature reduction can be calculating the norm of feature, or dot product of two feature vectors. The output of Feature Reduction should no longer have a index of feature.

Definition 3.4 (Transform Parameter)**.** The Parameters that are used to perform linear transformation of input Variable are Transform Parameters.

Here we need to emphasize that not only the weights of linear layers are considered as transform parameters, but also other weights such as weights in convolution, transposed convolution and bilinear layers are also transform parameters because operations can be converted into one or series of GEneralized Matrix Multiplications (GEMM).

Definition 3.5 (Transform Operator)**.** The Transform Operators are computational operators that perform linear transformation or operations can be converted into linear transformation on the feature dimension of the input.

Definition 3.6 (Transform Operator Split)**.** Transform operators in the computational graph can be split into two identical nodes. One copy connects with all edges going into the node, and another connects with all edges going out of the node. If there is a Transform Parameter involved in the computation, it should be split into Transform Parameter-in and Transform Parameter-out. The former should be connected to first duplication and the latter should be connected to second duplication.

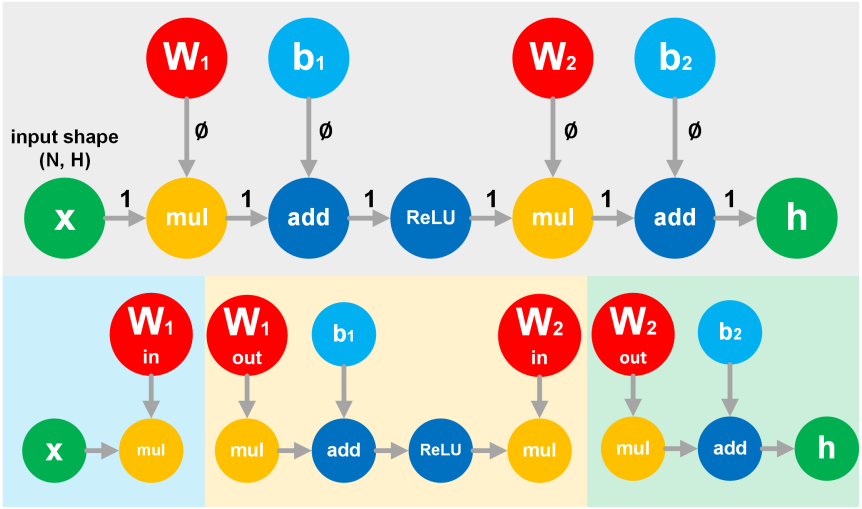
Algorithm 3.1: Split Computation Graph

Input: Graph $G = (V, E)$, Index of Feature in input $\mathbf{I} = \{i_0, i_1, \cdots, i_n\}$

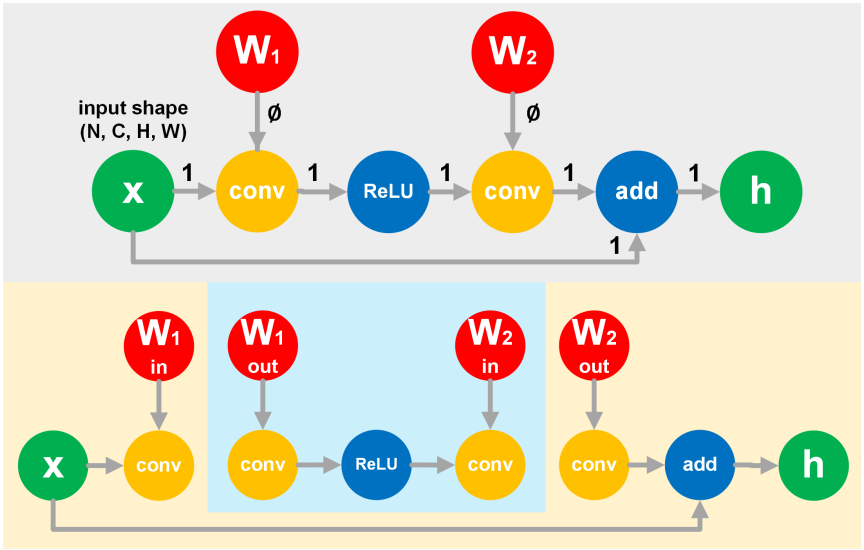
Initialization: an empty set L, an empty set S for all transform operators, a dictionary $indexOfFeature$

foreach $n \in$ Input Nodes **do**
  if n is a input **then**
    $indexOfFeature[n] \leftarrow \mathbf{I}[n]$
  end
  else if n is a Parameter **then**
    $indexOfFeature[n] \leftarrow \emptyset$
end
while There are unvisited nodes in G **do**
  foreach Node $v \in V$ **do**
    if v is not in L and all predecessors of v are in L **then**
      foreach Edge $(u, v) \in E$ **do**
        $curIndexOfFeatures \leftarrow$ the values from $indexOfFeature[u]$;
      end
      if v perform Feature Reduction given the $curIndexOfFeatures$ **then**
        $indexOfFeature[v] \leftarrow \emptyset$;
      else if v perform index manipulation on tensor **then**
        $indexOfFeature[v] \leftarrow$ perform manipulation on $curIndexOfFeatures$ according to v;
      else
        $indexOfFeature[v] \leftarrow \bigcup curIndexOfFeatures$;
        if v is a Transform Operator **then**
          Add v to S;
        end
      end
      Add v to L;
    end
  end
end
foreach $n \in S$ **do**
  Split G by Definition 3.6
end

Result: $\mathcal{G} = \{G_1, G_2, \cdots, G_n\}$, the split connected components of input G. $indexOfFeature$ the dict of Index of Feature of all nodes.



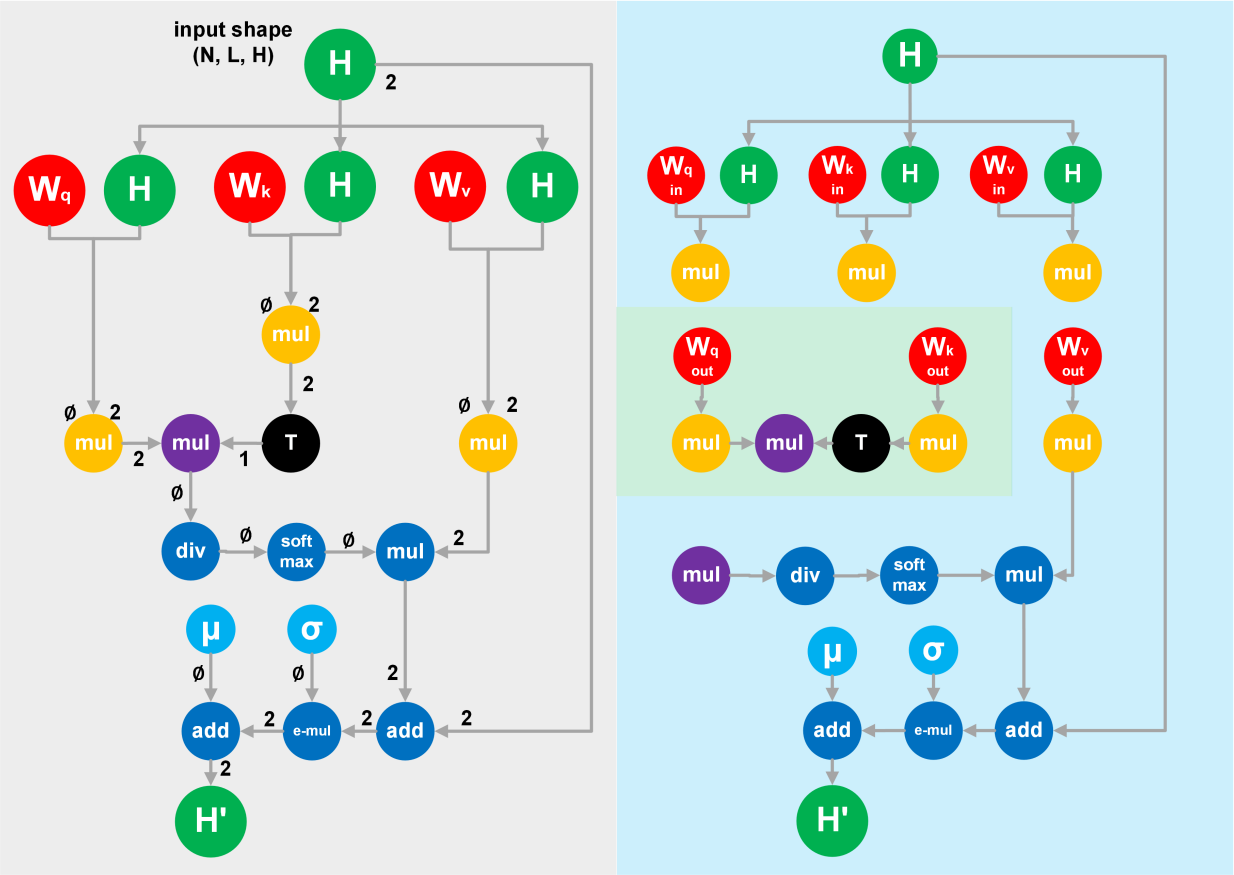
(a) MLP example of graph split



(b) Convolution network example of graph split

Figure 3.3: Examples of Transform Parameter splitting, different background color represent different groups. All inputs have batch size N. **orange** nodes are Transform Operators, **red** nodes are Transform Parameters, **purple** nodes are Feature Reduction nodes, **green** nodes are input, outputs and variables, **skyblue** nodes are other parameters, and **black** nodes are involved in tensor index manipulation. The number or $\emptyset$ attached to edges are $indexOfFeature$ of the node output. The number on the edges indicate the dimension of feature.

With Transform Operator Splitting, we are able to split the computational graph into multiple connected components using Algorithm 3.1 using the idea of "tracking the transformation on index of feature". The algorithm is based on topological sorting and of complexity $\mathcal{O}(|V|)$. Figure 3.3 illustrate three simple examples: multi-layer perception, convolution network with residual connection and self-attention with layer norm.



(c) Self-attention with norm example of graph split

Figure 3.3: Cont'd

After splitting the computation graph, the parameter groups can be gained by finding all parameters in one connected component. We can state that the parameter selection strategy of all Parameters involved in the same connected component can be decided by one binary mask. This is because in this sub-graph no transformation on feature is performed, and all manipulation of feature tensors is in the same linear space.

Algorithm 3.2: Parameter Group-Consistent Parameter Selection

Input: Split Graph $\mathcal{G} = \{G_0, G_1, \cdots, G_m\}$, Teacher Parameters $\mathcal{W} = \{W_0, W_1, \cdots, W_d\}$, Student Parameter Shape $\mathcal{S} = \{S_0, S_1, \cdots, S_d\}$

$\mathcal{G}_s \leftarrow$ Sort $\mathcal{G}$ by number of parameters in each G_i;

foreach $G_i \in \mathcal{G}_s$ **do**

  $T_t \leftarrow$ length of specified dimension of parameters involved in G_i;

  `// Here "specified" means "in" or "out" mentioned in graph split`

  $T_s \leftarrow$ length of specified dimension of parameters involved in G_i in student based on $\mathcal{S}$;

  $L \leftarrow$ empty list;

  for $i \leftarrow 0$ to T_t **do**

    $score \leftarrow 0$;

    foreach Parameter $p \in G_i$ **do**

      $W_p \leftarrow$ weight of teacher parameter correspond p in $\mathcal{W}$;

      $score \leftarrow score + \|W_p[\cdots, i, \cdots]\|_F^2$;

      `/* Here` $\|W_p[\cdots, i, \cdots]\|_F$ `means select the corresponding dimension of weight such as row, column, or channels */`

    end

    $score \leftarrow \sqrt{score}$ Add $score$ to L;

  end

  $indexToKeep \leftarrow$ topKIndex(L, T_s);

  foreach Parameter $p \in G_i$ **do**

    $W_p \leftarrow$ weight of teacher parameter correspond p in $\mathcal{W}$;

    $W_p' \leftarrow$ select the parameter to keep with $indexToKeep$;

    replace W_p with W_p' in $\mathcal{W}$;

  end

end

Result: Selected parameters $\mathcal{W}$ for student initialization

As a result, in our Mask2Former scenario, we only need to find a selection strategy for each sub-group or parameter group. And the only Transform Operators involved are linear layers and convolutional layers. For a linear layer, pruning on W_{in} means removing rows of the weight matrix, and pruning on W_{out} means removing columns of the weight matrix. For a convolutional layer, pruning on W_{in} means removing specified channels from all kernels and pruning on W_{out} means remove entire specified kernels.

To generate the pruning strategy, we use the simple l_2-norm of weights to measure

the importance of feature according to DepGraph. In contrast, DepGraph proposed using sparse training with a regularization term to find the pruning strategy, which is more time-consuming and more computation-intensive. Since we have known the target architecture and most of the parameters are grouped into two groups: image encoder group and mask decoder group, we proposed a heuristic algorithm that generates the parameter group consistent parameter selection strategy efficiently. Algorithm 3.2 illustrate the process the parameter group consistent parameter selection.

3.4 KNOWLEDGE DISTILLATION FOR PANOPTIC SEGMENTATION

Upon determining the initial parameters, it is essential to refine the teacher model through distillation to enhance performance. Existing distillation methodologies predominantly fall under two categories: prediction distillation and feature distillation. Our proposed distillation technique incorporates both approaches. Specifically, for prediction distillation, we employ an adaptive matching strategy, while for feature distillation, we utilize image feature distillation, deformable attention distillation, and a combination of self and cross-attention distillation.

3.4.1 Image Feature Distillation

Most previous research has focused on image feature distillation, and this is also a crucial part in panoptic segmentation. In this work, we adopt Masked Generative Distillation [50]. We randomly mask several feature pixels and then use a simple convolution layer to recover the masked feature. Formally, given the student feature S and teacher feature T, the image feature distillation loss can be

$$\mathcal{L}_{\text{feature}} = \sum_{l=1}^{L} \sum_{c=1}^{C} \sum_{h=1}^{H} \sum_{w=1}^{W} \text{SmoothL1} \left(T_{c,h,w}^{l}, \mathcal{G} \left(f_{\text{align}}(S_{c,h,w}^{l}) \cdot M_{c,h,w}^{l} \right) \right) \tag{3.1}$$

Where C, H, W, L represent channel, height, width and layer. f_{align} is used for matching the the dimension of the student and teacher, which a simple convolution layer. M is a binary random mask. $\mathcal{G}$ is the generator, which is a two-layer convolution network with ReLU activation.

3.4.2 Adaptive Matching for Prediction Distillation

The output of Mask2Former model is a set of binary masks and corresponding classes of

the thing or stuff, and can be represented as $\{\hat{\mathbf{y}}_i = (\mathbf{c_i}, \mathbf{m_i})\}_{i=0}^{N-1}$, where $\mathbf{m}_i$ is the mask, $\mathbf{c_i}$ is the predicted class, and N is the number of queries. A major problem on prediction distillation of panopatic distillation is that the output of model is unordered, so a matching between ground truth $\mathbf{y}_i$ and prediction $\hat{\mathbf{y}}_i$ is needed. Because of the same reason, an adaptive matching is required between teacher and student predictions. To perform the matching, we first calculate the cost of matching given two predictions of teacher $\mathbf{y^t} = (\mathbf{c^t}, \mathbf{m^t})$ and student $\mathbf{y^s} = (\mathbf{c^s}, \mathbf{m^s})$.

$$\mathcal{C}_{s-t}(\mathbf{y^s}, \mathbf{y^t}) = \alpha_c \text{BCE}(\mathbf{m^s}, \mathbf{m^t}) + \beta_c \text{KL-Div}(\mathbf{c^s}, \mathbf{c^t}) \tag{3.2}$$

Where α_c and β_c are hyper-parameters to balance the mask prediction and class prediction. We minimize the overall cost to obtain the the matching result σ

$$\sigma = \arg\min_{\sigma} \sum_{i=0}^{N-1} \mathcal{C}(\mathbf{y^s}_i, \mathbf{y^t}_{\sigma(i)}) \tag{3.3}$$

Based on the matching result we can decide the matching loss between the student and teacher.

$$\mathcal{L}_{\text{match}} = \sum_{i=0}^{N-1} \mathcal{C}(\mathbf{y^s}_i, \mathbf{y^t}_{\sigma(i)}) \tag{3.4}$$

3.4.3 Deformable Attention Distillation

The image encoder is mainly a transformer encoder with deformable attention. The deformable attention can be represented as

$$\text{DeformAttn}(\mathbf{x}) = \sum_{m=0}^{M-1} \mathbf{W}_m \left[\sum_{k=0}^{K-1} A_{mqk} \cdot \mathbf{W}'_m \mathbf{x}(\mathbf{p}_q + \Delta \mathbf{p}_{mqk}) \right] \tag{3.5}$$

$$\text{Where} \quad \Delta \mathbf{p} = \mathbf{W_p x}, \quad \mathbf{A} = \mathbf{W_A x}(\mathbf{p}_q + \Delta \mathbf{p}_{mqk}) \tag{3.6}$$

Here $\Delta \mathbf{p}$ is the sampling offset relative to the self position, and $\mathbf{A}$ is the attention weight. And these two variables decide the behavior of deformable attention. As a result, to distill deformable attention modules, we use loss

$$\mathcal{L}_{\text{deformable}} = \text{MSE}(\Delta \mathbf{p^s}, \Delta \mathbf{p^t}) + \text{MSE}(\mathbf{A^s}, \mathbf{A^t}) \tag{3.7}$$

to let the student mimic the teacher's deformable attention behavior.

3.4.4 Self-Attention and Cross-Attention Distillation

Attention score distillation is a simple but effective approach to transformer distillation. The mask embedding decoder of the model we worked on leverages the self-attention between queries and cross-attention between queries and image features. However, there is inconsistency between the teacher's attention map and student's attention map because of unordered outputs or different query sizes. So we need to employ the prediction matching result. Formally:

$$\mathcal{L}_{\text{attention}} = \alpha_a \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \left(\mathbf{A^s}_{\text{self}}[i,j] - \mathbf{A^t}_{\text{self}}[\sigma(i), \sigma(j)] \right)^2 + \tag{3.8}$$

$$\beta_a \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \left(\mathbf{A^s}_{\text{cross}}[i,j] - \mathbf{A^t}_{\text{cross}}[\sigma(i), j] \right)^2 \tag{3.9}$$

Here α_a and β_a are hyper-parameters that balance the two types of attention loss.

3.4.5 Layer Matching for Distillation

In the context of layerwise distillation, two scenarios arise. The first scenario is when the student and teacher have the same architecture and number of layers, allowing for a straightforward layer-to-layer correspondence. In contrast, the second scenario poses a challenge when the student and teacher differ in depth. To address this, we explored three approaches: First, Last, and Dilated, which are analogous to layer initialization techniques, as depicted in Figure 3.2. Also, it requires a different set of hyper-parameters. Our experiments shed light on the effectiveness of these methods in the context of distillation.

3.4.6 Distillation Loss

We combine all the losses mentioned above for distillation and the original training objective $\mathcal{L}_{\text{train}} = a\mathcal{L}_{\text{dice}} + b\mathcal{L}_{\text{BCE}} + c\mathcal{L}_{\text{cls}}$ of Mask2Former.

$$\mathcal{L}_{\text{full}} = \sum_{i=0}^{L_d-1} (a\mathcal{L}_{\text{dice}} + b\mathcal{L}_{\text{BCE}} + c\mathcal{L}_{\text{cls}} + d\mathcal{L}_{\text{match}} + e\mathcal{L}_{\text{attention}}) + \sum_{j=0}^{L_e-1} (f\mathcal{L}_{\text{deformable}} + g\mathcal{L}_{\text{feature}}) \tag{3.10}$$

Here L_e and L_d are number of encoder and decoder layers of the student.

3.4.7 Multi-Stage Distillation

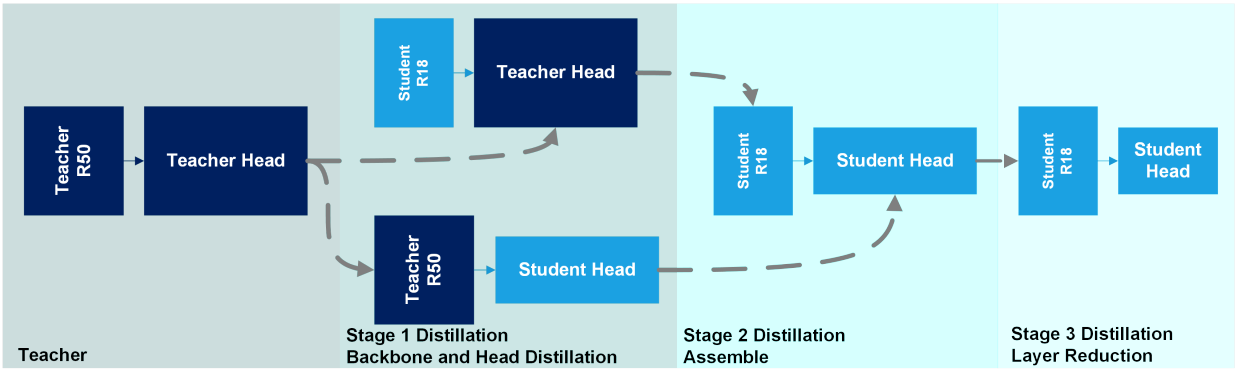


Figure 3.4: Multi-stage distillation. Given the teacher and student, three stage of distillation is performed. First stage is to distill backbone and segmentation head in a separate way. Second stage is to adapt the distilled student head to distilled student backbone. Final step is to further reduce the student layer.

Typically, knowledge distillation aims at aligning the feature and prediction of the teacher and student based on the numerical difference but ignores the functional difference of different components of panoptic segmentation model. As a result, we proposed a multi-stage distillation to make the different components functionally consistent as Figure 3.4 illustrated. The first stage disassembles the student and teacher, and trains the student's head with teacher's backbone and the student's backbone with teacher's head. With this approach, we can make the backbone and segmentation head more functionally consistent with the teacher, and yield a better performance.

CHAPTER 4: EXPERIMENTS

4.1 DATASET

In this study, we utilized the MS COCO [45] dataset, a widely recognized benchmark for various computer vision tasks, including object detection, instance segmentation, image captioning, and panoptic segmentation. Specifically, for the panoptic segmentation task, the dataset comprises 163,957 labeled images, with annotations for 80 thing classes and 53 stuff classes.

4.2 IMPLEMENTATION DETAILS

All experiments are conducted with 4 NVIDIA A40 or 4 NVIDIA A100 using MMDetection [51] framework. The general hyper-parameters used for training are in the table.

Name	Value	
schedule	1x	3x
decay epochs	8/11	28/34
epochs	12	36
backbone learning rate	1e-4	
head learning rate	1e-5	
decay gamma	0.1	
weight decay	0.05	
optimizer	Adam	
Adam eps	1e-8	
Adam beta	(0.9, 0.999)	

Table 4.1: Training Hyperparameters

Furthermore, we have different schemes of distillation for layer-consistent and layer-inconsistent distillation. Here layer-consistent means the teacher and student have exactly the same number of layers while layer-inconsistent has different numbers.

Name	a	b	c	d	e	f	g	α_c	β_c	α_a	β_a
Layer-consistent	2.0	5.0	5.0	1.0	1.0	1.0	2e-6	10.0	2.0	0.125	1.0
Layer-inconsistent	2.0	5.0	5.0	1.0	0.0	0.0	4e-6	4.0	0.8	0.0	0.0

Table 4.2: Hyper-parameters used for distillation for different distillation scenarios. Here some values are zero, which means we did not use this loss in the corresponding scenario.

4.3 METRICS

We assessed the performance of our model using the widely used **PQ** (Panoptic Quality) metrics. PQ is a composite score that can be obtained by multiplying the scores of two metrics: **SQ** segmentation quality and **RQ** recognition quality. In addition, we evaluated the efficiency of our model by measuring its FLOPS and number of parameters.

4.4 BASELINES

We compare our models with previous baselines. These baselines include previous CNN-based and Transformer-based models.

Panoptic FPN [1] is a modifed version of Mask-RCNN with a novel Feature Pyrimaid Network architecture for both instance segmentation and semantic segmentation.

UPSNet [2] is a unified panoptic segmentation network, and has three heads: a semantic segmentation head, an instance segmentation head, and a parameter-free panoptic head, which solve the subtasks simultaneously.

MaX-DeepLab [9] is an end-to-end model for panoptic segmentation that simplifies the previous pipeline by directly predicting class-labeled masks with a mask transformer and training with a panoptic quality inspired loss via bipartite matching.

DETR [8] is previously used in object detection, and can also be used to perform panoptic segmentation. So we also employ DETR as a baseline.

Panoptic Segformer [12] is a transformer-based segmentation model which contains three innovative components: an efficient deeply-supervised mask decoder, a query decoupling strategy, and an improved post-processing method.

MaskFormer [10] is a simple mask classification model which predicts a set of binary masks, each associated with a single global class label prediction.

Mask DINO [13] extends DINO (DETR with Improved Denoising Anchor Boxes) by adding a mask prediction branch which supports all image segmentation tasks (instance, panoptic, and semantic).

To demonstrate the effectiveness of our approach, we compare three variants of our proposed model using ResNet-50 and ResNet-18 backbone with previous baseline. **Mask2Former-S** is based on the teacher model Mask2Former but with $L_e = 3$ and $L_d = 3$. **Mask2Former-T** has the same number of layers with Mask2Former but with embedding size $d = 128$. **Mask2Former-ST** has $L_e = 3$, $L_d = 3$ and $d = 128$, which minimizes the number of parameters and computation. We also applied our multi-stage distillation to build our smallest model, Mask2Former-ST with ResNet-18 backbone.

4.5 MAIN RESULTS

Model	Backbone	Input Res	Epochs	#Query	PQ	PQ^{th}	PQ^{st}	Params (M)	Flops (G)
Panoptic FPN	R50	800 × 1333	12	Dense	40.9	48.3	29.7	46.1	-
Panoptic FPN	R50	800 × 1333	36	Dense	42.5	50.3	30.7	46.1	-
UPSNet	R50	800 × 1333	36	Dense	42.5	48.5	33.4	-	-
Panoptic-DeepLab	Xception-71	1025 × 1025	216	Dense	41.2	44.9	35.7	46.7	274
MaX-DeepLab-S	MaX-S	1025 × 1025	216	128	48.4	53.0	41.5	61.9	324
MaX-DeepLab-L	MaX-L	1025 × 1025	216	128	51.1	57.0	42.2	451	3692
DETR	R50	1024 × 1024	325	300	43.4	48.2	36.3	42.9	248
Panoptic Segformer	R50	800 × 1333	24	300	49.6	54.4	42.4	51.0	214
MaskFormer	R50	1024 × 1024	300	100	46.5	51.0	39.8	45.0	181
Mask2Former	R50	1024 × 1024	50	100	51.9	57.7	43.0	44.0	226
MaskDINO	R50	1024 × 1024	50	100	<u>52.3</u>	<u>58.3</u>	<u>43.2</u>	52.0	280
Mask2Former-S	R50	1024 × 1024	12	100	49.5	55.0	41.1	32.3	169
Mask2Former-S	R50	1024 × 1024	36	100	**50.4**	**56.0**	**41.9**	**32.3**	**169**
Mask2Former-T	R50	1024 × 1024	12	100	49.3	54.8	41.2	29.8	136
Mask2Former-T	R50	1024 × 1024	36	100	**50.8**	**56.5**	**42.1**	**29.8**	**136**
Mask2Former-ST	R50	1024 × 1024	12	100	47.2	52.3	39.5	26.4	111
Mask2Former-ST	R50	1024 × 1024	36	100	**48.7**	**54.2**	**40.5**	**26.4**	**111**
Mask2Former-ST	R18	1024 × 1024	12	100	**41.5**	**45.5**	**35.5**	**13.7**	**68**

Table 4.3: Main Result, the **bold** result indicates the best result of our model under specific layer, hidden dimension and backbone configuration. The <u>underlined</u> results indicate the state-of-the-art result.

From the result, we can see our proposed "shallower" model, Mask2Former-S, and "thinner" model, Mask2Former-T, maintain competitive performance compared to previous baselines while requiring fewer parameters and less computation. Furthermore, our Mask2Former-ST model with a ResNet-18 backbone, outperforms previous state-of-the-art models such as Panoptic FPN and Panoptic-DeepLab with a smaller number of parameters and flops.

4.6 ABLATION STUDY ON INITIALIZATION

The method that initializes the student model parameters will influence the result.

In this part, we will discuss how layer initialization and our parameter group-consistent parameter selection improve the performance.

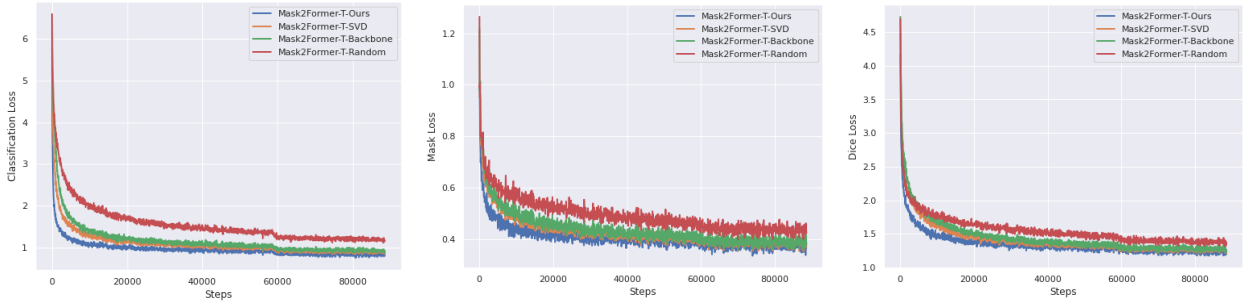
4.6.1 Layer Initialization

Name	PQ	PQ^{th}	PQ^{st}
Mask2Former-S First	**49.1**	**54.5**	**40.9**
Mask2Former-S Last	48.6	54.0	40.5
Mask2Former-S Dilated	48.6	54.1	40.4

Table 4.4: Result of different layer initialization scheme.

To decide the best initialization strategies, we performed extensive experiment on three different initialization strategy. From the result of Table 4.4, we found that using the first several layers to initialize the student model can reach the best performance, and we keep this initialization strategy for remaining experiments.

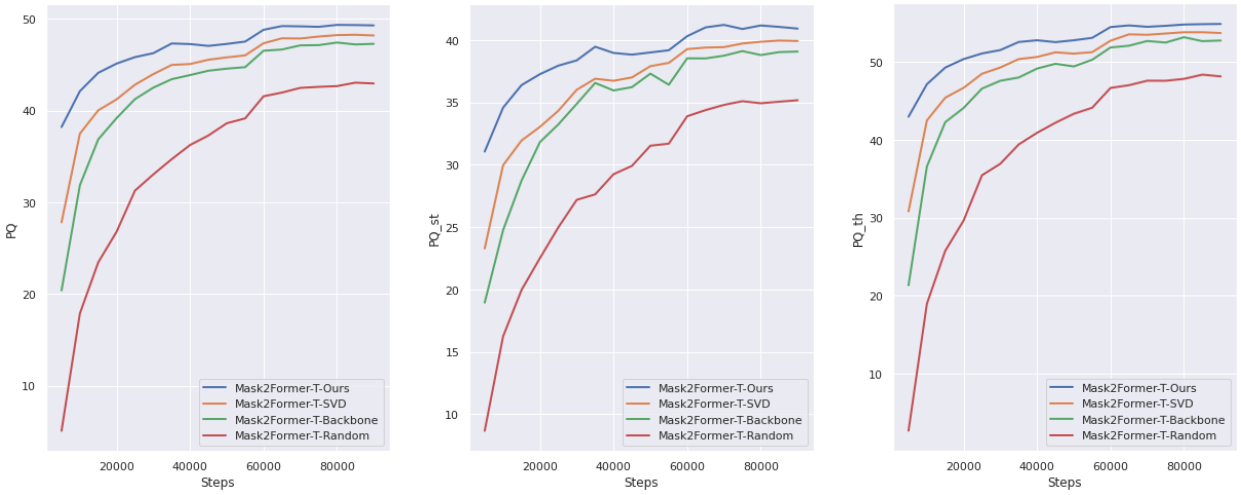
4.6.2 Model Initialization Parameter Selection



(a) Train Classification Loss (b) Train Mask BCE Loss (c) Train Mask Dice Loss

Figure 4.1: The train loss of different parameter initialization strategy.

To demonstrate the effectiveness our proposed method of group-consistent parameter selection, we compared the performance of model by randomly initializing the model (Mask2Former-T-Random), only inheriting the teacher's backbone (Mask2Former-T-Backbone), initializing the model via pruning model using SVD of the teacher's parameter, and our approach (Mask2Former-T-Ours). Figure 4.1 shows the decreasing of three training losses, and the result shows the proposed method converges faster than all other baselines during training.



(a) PQ result (b) PQ^{st} result (c) PQ^{th} result

Figure 4.2: The validation result of different parameter initialization strategy.

Figure 4.2 show the validation PQ, PQ^{th} and PQ^{st} results of 1x training on MS COCO dataset at different training steps.

From the results, we can see just initializing the backbone can significantly improve the performance. Using the typical approach SVD also improves the performance. But our group-consistent parameter selection approach inherits the teacher model's performance best among all these approaches. Table 4.5 shows the final results of different parameter selection strategies, and the proposed approach out-performed the backbone initialization only approach by 1.93 PQ, and the SVD approach by 1.08 PQ.

Name	PQ	PQ^{th}	PQ^{st}
Mask2Former-T Random	42.96	48.12	35.17
Mask2Former-T Backbone	47.42	53.14	38.79
Mask2Former-T SVD	48.27	53.78	39.96
Mask2Former-T Ours	**49.35**	**54.77**	**41.16**

Table 4.5: Experimental results of different parameter initialization approaches.

4.7 ABLATION STUDY ON KNOWLEDGE DISTILLATION

To evaluate the effectiveness of different distillation losses, layer matching strategies and multi-stage distillation, we performed ablation study on different distillation losses using Mask2Former-T and layer matching strategies using Mask2Former-S.

4.7.1 Knowledge Distillation Loss

I.Prediction	II.Deformable	III.Image Feature	IV.Attention	PQ	PQ^{st}	PQ^{th}
				45.80	51.23	37.59
✓				46.55	51.63	38.88
✓	✓			46.99	52.59	38.55
✓	✓	✓		47.25	52.74	**38.95**
✓	✓	✓	✓	**47.42**	**53.14**	38.79

Table 4.6: Results of using different combination of distillation losses

From Table 4.6, we can conclude each loss involved in the distillation benefits the final performance. The adaptive matching-based loss improves the performance by 0.75 PQ, deformable attention loss improves by another 0.44 PQ, image feature loss improves by additional 0.26 and attention loss brings 0.17 PQ gain finally.

4.7.2 Layer Matching Strategy

To figure out how to match the teacher and student's layers properly so that we can maximize the distillation performance. We performed experiment on different matching strategy. Based on previous initialization result, we initialize our Mask2Former-S model with "first" strategy from the teacher.

Name	PQ	PQ^{th}	PQ^{st}
Mask2Former-S First	49.35	54.77	41.16
Mask2Former-S Last	**49.53**	**55.00**	**41.27**
Mask2Former-S Dilated	49.03	54.22	41.19

Table 4.7: Result of different layer matching scheme

We perform distillation using only prediction distillation and experiment on three distillation matching strategies. The result shows that the "Last" strategy is the best for distillation, which is because the last several layers provides most refined feature so that the student can learn from it.

4.7.3 Multiple-Stage Distillation

To demonstrate the effectiveness of multiple stage distillation works better than direct distillation, we performed ablation study. We compared the result with a longer 3x training

schedule Mask2Former-R18-3x. Another baseline to compare is to apply our distillation approach for 3x schedule training, Mask2Former-R18-distill-3x. The result we showed in the main table is a multi-stage distillation, which follows the path illustrate in Figure 3.4.

Model	PQ	PQ^{st}	PQ^{th}
Mask2Former-R18-3x	40.17	44.53	33.58
Mask2Former-R18-distill-3x	41.06	44.73	35.52
Mask2Former-R18-multi-distill-3x	**41.48**	**45.47**	**35.46**

Table 4.8: Ablation study result on multi-stage distillation.

The experimental results show that our multi-stage distillation based on the idea of functionality consistency out-performs the baseline by 1.29 PQ. In the meantime, our proposed multi-stage distillation also outperforms distillation using a longer training schedule by 0.42 PQ, which demonstrates the effectiveness of our multi-stage approach.

CHAPTER 5: RELATED WORKS

5.1 KNOWLEDGE DISTILLATION FOR TRANSFORMER-BASED DETECTION

Although there is no previous work in Transformer-based panoptic segmentation, some of the previous works focusing on transformer-based detector are related to this work.

5.1.1 DETRDistill

DETRDistill [41] first performs a sparse matching paradigm with progressive stage-by-stage instance distillation. And then it uses an attention-agnostic feature distillation module to overcome the ineffectiveness of conventional feature imitation. Finally the author introduced teacher-assisted assignment distillation. This work has similar matching based prediction distillation and attention distillation with us, but they did not mention the distillation of head and did not research on how to directly initialize from the teacher's parameter.

5.1.2 D^3ETR

D^3ETR [43] focused on distillation using transformer decoder. The author proposed MixMatching, which is a combination of adaptive matching and fixed matching. The author applied this MixMatching on DETR and conditional DETR and achieved competitive performance. However, in this work, the authors only focus on using a smaller backbone and performing distillation on the decoder.

5.1.3 KD-DETR

KD-DETR [42] decoupled detection and distillation tasks by introducing a set of specialized object queries to construct distillation points, and then further proposed a general-to-specific distillation point sampling strategy. The experiments on DAB-DETR, Deformable-DETR, and DINO demonstrate the effectiveness of the approach. However, they also did not explore on how to reduce the decode itself.

5.2 STRUCTURAL SPARSITY CONSISTENT PRUNING

DepGraph [49] proposed consistent structural sparsity constrained model pruning based on the parameter dependency, which is a similar approach with our proposed approach when

selecting the parameter for student initialization. However, DepGraph fails to generalize to more complicated models that include novel components such as deformable attention, and it also requires a sparse training to get the pruning strategy. The approach we proposed is based on computation graph and can generalize to more models, and we used an heuristic algorithm to select parameter instead of sparse training, which is simpler than sparse training but effective.

CHAPTER 6: CONCLUSIONS

6.1 CONCLUSIONS

In this work, we propose a series of approaches to distill a transformer-based panoptic segmentation model, Mask2Former. The approaches include layer initialization, group consistent parameter selection for initialization. We further explore the knowledge distillation on different components of the model and produce models whose performance out-performs multiple strong previous baselines with far fewer parameters. Our study also indicates the potential of transformer-based panoptic segmentation, and it is possible to design models with smaller heads and backbones while maintaining competitive performance.

6.2 LIMITATION AND FUTURE WORKS

In this work, the experiment is only limited to Mask2Former, but our approaches can be extended to other strong transformer-based panoptic segmentation model such as Mask DINO [13] and kMaX-DeepLab [52]. Future work can generalize our approach to those aforementioned models. In the meantime, our distillation approach requires more training time than training the model itself because it requires the inference of the teacher model. Furthermore, our approach can not generalize to other architectures such as diffusion based panoptic segmentation. Recent diffusion-based panoptic segmentation models such as Pix2Seq-$\mathcal{D}$ [53] also reached competitive performance, and it remains an open question how to build efficient diffusion-based panoptic segmentation models.

REFERENCES

[1] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9404–9413.

[2] Y. Xiong, R. Liao, H. Zhao, R. Hu, M. Bai, E. Yumer, and R. Urtasun, "Upsnet: A unified panoptic segmentation network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8818–8826.

[3] S. Qiao, L.-C. Chen, and A. Yuille, "Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 10 213–10 224.

[4] R. Mohan and A. Valada, "Efficientps: Efficient panoptic segmentation," *International Journal of Computer Vision*, vol. 129, pp. 1551 – 1579, 2020.

[5] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[7] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell et al., "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[8] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*. Springer, 2020, pp. 213–229.

[9] H. Wang, Y. Zhu, H. Adam, A. Yuille, and L.-C. Chen, "Max-deeplab: End-to-end panoptic segmentation with mask transformers," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 5463–5474.

[10] B. Cheng, A. Schwing, and A. Kirillov, "Per-pixel classification is not all you need for semantic segmentation," *Advances in Neural Information Processing Systems*, vol. 34, pp. 17 864–17 875, 2021.

[11] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, "Masked-attention mask transformer for universal image segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1290–1299.

[12] Z. Li, W. Wang, E. Xie, Z. Yu, A. Anandkumar, J. M. Alvarez, P. Luo, and T. Lu, "Panoptic segformer: Delving deeper into panoptic segmentation with transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1280–1289.

[13] F. Li, H. Zhang, S. Liu, L. Zhang, L. M. Ni, H.-Y. Shum et al., "Mask dino: Towards a unified transformer-based framework for object detection and segmentation," *arXiv preprint arXiv:2206.02777*, 2022.

[14] 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*, 2021, pp. 9650–9660.

[15] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[16] P. Zhang, Z. Kang, T. Yang, X. Zhang, N. Zheng, and J. Sun, "Lgd: label-guided self-distillation for object detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 3, 2022, pp. 3309–3317.

[17] G. Li, X. Li, Y. Wang, S. Zhang, Y. Wu, and D. Liang, "Knowledge distillation for object detection via rank mimicking and prediction-guided feature imitation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 2, 2022, pp. 1306–1313.

[18] Z. Zheng, R. Ye, P. Wang, D. Ren, W. Zuo, Q. Hou, and M.-M. Cheng, "Localization distillation for dense object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 9407–9416.

[19] H. Zhou, Z. Ge, S. Liu, W. Mao, Z. Li, H. Yu, and J. Sun, "Dense teacher: Dense pseudo-labels for semi-supervised object detection," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part IX*. Springer, 2022, pp. 35–50.

[20] Z. Yang, Z. Li, X. Jiang, Y. Gong, Z. Yuan, D. Zhao, and C. Yuan, "Focal and global knowledge distillation for detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4643–4652.

[21] L. Zhang and K. Ma, "Improve object detection with feature-based knowledge distillation: Towards accurate and efficient detectors," in *International Conference on Learning Representations*, 2021.

[22] Y. Liu, K. Chen, C. Liu, Z. Qin, Z. Luo, and J. Wang, "Structured knowledge distillation for semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2604–2613.

[23] Y. Hou, X. Zhu, Y. Ma, C. C. Loy, and Y. Li, "Point-to-voxel knowledge distillation for lidar semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8479–8488.

[24] C. Yang, H. Zhou, Z. An, X. Jiang, Y. Xu, and Q. Zhang, "Cross-image relational knowledge distillation for semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 319–12 328.

[25] S. Chen, Z. Hong, G.-S. Xie, W. Yang, Q. Peng, K. Wang, J. Zhao, and X. You, "Msdn: Mutually semantic distillation network for zero-shot learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 7612–7621.

[26] Y. Tian, D. Krishnan, and P. Isola, "Contrastive representation distillation," *arXiv preprint arXiv:1910.10699*, 2019.

[27] W. Park, D. Kim, Y. Lu, and M. Cho, "Relational knowledge distillation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3967–3976.

[28] B. Heo, J. Kim, S. Yun, H. Park, N. Kwak, and J. Y. Choi, "A comprehensive overhaul of feature distillation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1921–1930.

[29] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," *arXiv preprint arXiv:1612.03928*, 2016.

[30] H. Zhou, L. Song, J. Chen, Y. Zhou, G. Wang, J. Yuan, and Q. Zhang, "Rethinking soft labels for knowledge distillation: A bias-variance tradeoff perspective," *arXiv preprint arXiv:2102.00650*, 2021.

[31] C. Shu, Y. Liu, J. Gao, Z. Yan, and C. Shen, "Channel-wise knowledge distillation for dense prediction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5311–5320.

[32] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5776–5788, 2020.

[33] W. Wang, H. Bao, S. Huang, L. Dong, and F. Wei, "MiniLMv2: Multi-head self-attention relation distillation for compressing pretrained transformers," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Online: Association for Computational Linguistics, Aug. 2021. [Online]. Available: https://aclanthology.org/2021.findings-acl.188 pp. 2140–2151.

[34] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[35] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, "Mobilebert: a compact task-agnostic bert for resource-limited devices," *arXiv preprint arXiv:2004.02984*, 2020.

[36] X. Chen, Q. Cao, Y. Zhong, J. Zhang, S. Gao, and D. Tao, "Dearkd: data-efficient early knowledge distillation for vision transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 052–12 062.

[37] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International conference on machine learning*. PMLR, 2021, pp. 10 347–10 357.

[38] D. Jia, K. Han, Y. Wang, Y. Tang, J. Guo, C. Zhang, and D. Tao, "Efficient vision transformers via fine-grained manifold distillation," *arXiv preprint arXiv:2107.01378*, 2021.

[39] J. Liu, B. Liu, H. Li, and Y. Liu, "Meta knowledge distillation," *arXiv preprint arXiv:2202.07940*, 2022.

[40] Z. Yang, Z. Li, A. Zeng, Z. Li, C. Yuan, and Y. Li, "Vitkd: Practical guidelines for vit feature knowledge distillation," *arXiv preprint arXiv:2209.02432*, 2022.

[41] J. Chang, S. Wang, G. Xu, Z. Chen, C. Yang, and F. Zhao, "Detrdistill: A simple knowledge distillation framework for detr-families."

[42] Y. Wang, X. Li, S. Wen, F. Yang, W. Zhang, G. Zhang, H. Feng, J. Han, and E. Ding, "Knowledge distillation for detection transformer with consistent distillation points sampling," *arXiv preprint arXiv:2211.08071*, 2022.

[43] X. Chen, J. Chen, Y. Liu, and G. Zeng, "D ̈3 etr: Decoder distillation for detection transformer," *arXiv preprint arXiv:2211.09768*, 2022.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[45] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Doll'a r, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[46] 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.

[47] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[48] Z. Xia, X. Pan, S. Song, L. E. Li, and G. Huang, "Vision transformer with deformable attention," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 4794–4803.

[49] G. Fang, X. Ma, M. Song, M. B. Mi, and X. Wang, "Depgraph: Towards any structural pruning," *arXiv preprint arXiv:2301.12900*, 2023.

[50] Z. Yang, Z. Li, M. Shao, D. Shi, Z. Yuan, and C. Yuan, "Masked generative distillation," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XI*. Springer, 2022, pp. 53–69.

[51] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu et al., "Mmdetection: Open mmlab detection toolbox and benchmark," *arXiv preprint arXiv:1906.07155*, 2019.

[52] Q. Yu, H. Wang, S. Qiao, M. Collins, Y. Zhu, H. Adam, A. Yuille, and L.-C. Chen, "k-means mask transformer," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIX*. Springer, 2022, pp. 288–307.

[53] T. Chen, L. Li, S. Saxena, G. Hinton, and D. J. Fleet, "A generalist framework for panoptic segmentation of images and videos," *arXiv preprint arXiv:2210.06366*, 2022.