DSFlash: comprehensive panoptic scene graph generation in realtime

Julian Lorenz, Vladyslav Kovganko, Elias Kohout, Mrunmai Phatak, Daniel Kienzle, Rainer Lienhart

Angaben zur Veröffentlichung / Publication details:

Lorenz, Julian, Vladyslav Kovganko, Elias Kohout, Mrunmai Phatak, Daniel Kienzle, and Rainer Lienhart. 2026. "DSFlash: comprehensive panoptic scene graph generation in realtime." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2026), June 3-7, 2026, Denver, CO, USA*, 17388–98. Piscataway, NJ: IEEE. https://doi.org/10.48550/arXiv.2603.10538.

DSFlash: Comprehensive Panoptic Scene Graph Generation in Realtime

Julian Lorenz Vladyslav Kovganko Elias Kohout
Mrunmai Phatak Daniel Kienzle Rainer Lienhart
University of Augsburg
julian.lorenz@uni-a.de

Abstract

Scene Graph Generation (SGG) aims to extract a detailed graph structure from an image, a representation that holds significant promise as a robust intermediate step for complex downstream tasks like reasoning for embodied agents. However, practical deployment in real-world applications - especially on resource constrained edge devices - requires speed and resource efficiency, challenges that have received limited attention in existing research. To bridge this gap, we introduce DSFlash, a low-latency model for panoptic scene graph generation designed to overcome these limitations. DSFlash can efficiently process video streams without compromising performance against existing state-of-the-art methods, with smaller DSFlash variants achieving 56 frames per second on a standard RTX 3090 GPU. Crucially, unlike prior approaches that often restrict themselves to salient relationships, DSFlash computes comprehensive scene graphs, offering richer contextual information while maintaining its superior latency. Furthermore, DSFlash is light on resources, requiring less than 24 hours to train on a single, nine-year-old GTX 1080 GPU. This accessibility makes DSFlash particularly well-suited for researchers and practitioners operating with limited computational resources, empowering them to adapt and fine-tune SGG models for specialized applications.

1. Introduction

Visual scene understanding has long been a core challenge in computer vision. One increasingly valuable approach involves scene graph generation (SGG) [19], which provides a rich, structured representation of an image. A scene graph is a set of nodes (representing instances in the scene) and edges (representing the relationships between them). This structure allows models to move beyond simple object detection towards comprehending complex interactions and context within a visual scene. When talking about relations, we refer to a triplet of (subject, predicate, object) that describes a directed relationship, *e.g.* ("person", "sitting on", "chair").

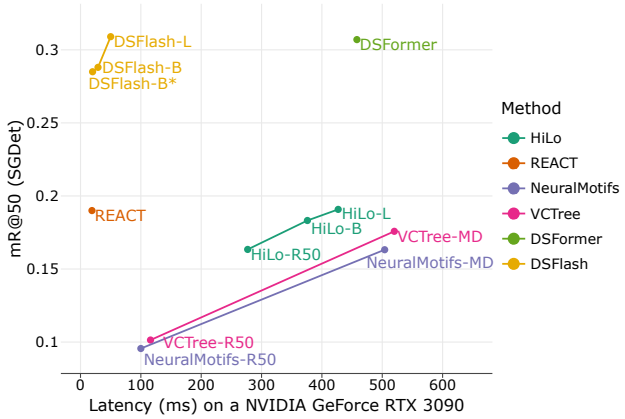


Figure 1. Performance comparison between our approach and previous work in terms of performance (*mR@50*) and latency (ms) on the PSG dataset [51].

Scene graphs have proven to be highly beneficial for numerous downstream tasks, including visual question answering [7, 16, 27], sports analysis [14, 37, 50], assisting surgery [15, 41], image captioning [35, 49, 52], and embodied reasoning tasks [12, 12, 33]. Crucially, they serve as an explainable intermediate step, offering clear, human-readable insights into a model's decision-making process, contributing a significant advantage over opaque end-to-end models.

SGG methods offer a lightweight and interpretable alternative to complex vision-language models (VLMs) in domains where structured scene understanding is prioritized over broad multimodal reasoning. However, despite their utility, the vast majority of research in this field has focused on improving the quality of extracted scene graphs, often at the expense of computational efficiency. Consequently, almost no research has been dedicated to low-latency scene graph generation models suitable for real-time or resource-constrained applications, with a few recent exceptions [34]. This gap is critical, as many real-world applications, especially autonomous systems with limited communication, demand efficient, scalable solutions that can operate under

strict latency constraints. To the best of our knowledge, there is no active research on low-latency models that are specifically targeted for panoptic scene graph generation (PSGG).

In this paper, we address this gap by presenting a novel, low-latency PSGG model, called DSFlash. Our work aims to demonstrate that highly efficient PSGG is still achievable while maintaining competitive performance. In fact, we go one step further and predict comprehensive scene graphs, *i.e.* for each image, DSFlash localizes all instances and classifies all potential relations between all instances. Even then, DSFlash runs faster than existing SGG methods that only classify a subset of relations.

Evaluation scenario. To benchmark SGG models for real-world applications, we evaluate model latency with a batch size of 1, simulating a constant feed of frames from a video stream. For our time measurements, we analyze latency as the time taken for a forward pass through a model. Since we are also interested in resource-constrained applications, we evaluate DSFlash not only using highly optimized hardware but also using older GPUs.

Our contributions can be summarized as:

1. We introduce DSFlash, a low-latency panoptic scene graph generation method with SOTA performance.
2. We propose a bidirectional relation predictor that halves the number of forward passes through the model head.
3. We present a mask-based dynamic patch pruning technique to reduce the number of processed tokens with minimal overhead.
4. We perform a thorough comparison with other PSGG models regarding performance and latency.
5. We show an extensive ablation study of the influence of DSFlash's various components on latency and performance.

Our code and model checkpoints can be found here: https://lorjul.github.io/dsflash

2. Related Work

In this section, we discuss related work regarding other SGG methods and relevant components that are employed in DSFlash.

2.1. Panoptic Scene Graph Generation

Contrary to traditional scene graph generation, panoptic scene graph generation (PSGG) [51] uses segmentation masks instead of bounding boxes to localize instances in a scene. Integrating segmentation masks into the training process gives a model additional context and understanding of various relations as opposed to bounding boxes. The most commonly used dataset for PSGG is the PSG dataset [51] which was created from the overlap of the Visual Genome [22] and COCO [28] datasets, resulting in 49k images with panoptic segmentation masks and scene graph annotations. Contrary to Visual Genome, PSG reduces the large predicate vocabulary of Visual Genome to 56 carefully selected classes.

The PSGG task has been tackled with various approaches, including methods that were originally built for Visual Genome like NeuralMotifs [53] or VCTree [44] but also specialized methods like HiLo [56] or DSFormer [31].

Similarly to classical SGG, PSGG methods can be categorized into two-stage and one-stage methods. Two-stage methods first predict instance locations and categories and use that information to perform the actual relation classification. One-stage methods do not separate these steps and directly predict a set of subject-predict-object triplets. Although recent research frequently argues for one-stage methods [51, 55, 56] because of their supposedly superior performance and efficiency, there have been recent publications opposing this claim [31, 34].

We choose to follow a two-stage approach for DSFlash since it possesses beneficial properties that align with our target setting. By reusing a pretrained segmentation backbone, DSFlash can be trained much quicker and with less resource requirements. On the other hand, if enough computational resources are available, the segmentation stage can be fine-tuned to specific datasets, even using potential pretraining recipies from adjacent domains.

2.2. Realtime Scene Graph Generation

Research on low-latency models for scene graph generation has remained largely overlooked, with only a few studies reporting their latency metrics.

However, Neau et al. [34] have begun to investigate common bottlenecks in recent SGG architectures. They evaluate multiple existing two-stage architectures [44, 45, 53] and optimize them for speed, achieving groundbreaking latency with good SGG performance using their new architecture, named REACT [34]. The most impactful contribution to the low latency is the replacement of the Faster-RCNN [13] backbones with YOLOv8 [18]. Additionally the authors propose to filter the number of box proposals before classifying relations between them, reducing the number of required relation predictions.

Although certainly a fundamental first step towards realtime efficient SGG models, the PSGG setting is only covered very briefly. In this work, we present a PSGG model that fully utilizes all available modalities and outperforms REACT [34] by a large margin in latency and PSGG performance.

2.3. DSFormer

To design DSFlash, we borrow some concepts from DSFormer [31] but replace many parts for increased performance and speed. The core idea behind DSFormer is to strictly decouple instance segmentation and relation predic-

tion not only into two separate stages of the network but two entirely separate networks. This enables DSFormer to immediately use any modern segmentation model in a plug-and-play fashion without having to retrain the scene graph model. Although certainly appealing, having to use two networks with separate backbones involves an immense resource inefficiency which we address with DSFlash.

To produce a panoptic scene graph, DSFormer first executes a segmentation model to retrieve segmentation masks. Next, for each possible combination of two segmentation masks, the second stage is prompted with the image, a subject segmentation mask, and an object segmentation mask. The model then classifies the relation between subject and object. To encode the location of subject and object, DSFormer adds a mask embedding to each patch:

$$token = patch + r_s \cdot t_s + r_o \cdot t_o + (1 - r_s - r_o) \cdot t_{bg}, \quad (1)$$

where r_s and r_o are the proportions of the patch area that is covered by the subject and object mask. t_s, t_o, t_{bg} are learnable tokens. Next, the enriched patches are processed by a set of transformer blocks. Finally, the relation head projects the tokens to the desired relation output.

While DSFormer achieves SOTA results on PSG, efficiency has been completely neglected by the authors. With DSFlash we borrow some core concepts to achieve similar PSGG scores while achieving significantly reduced inference time.

2.4. Fast Vision Backbones

As with most neural network architectures that target fast inference, an optimized backbone is key. A common choice is a network from the YOLO family [38], with recent iterations [17, 46, 48] achieving around 40 mAP on bounding box detection in less than 2 ms on a T4 GPU.

Although initially regarded as less efficient than convolutional neural networks, transformer based architectures, especially ViT-based methods have eventually shown superior computational efficiency [40] powered by highly optimized algorithms like flash attention [8, 9]. Especially during inference, ViTs benefit from data locality and use of optimized matrix operations. The recently published RF-DETR [39] model applies these insights. It is based on a transformer architecture, namely LW-DETR [43] and Deformable DETR [57] and achieves 48 mAP on bounding box detection in 2.3 ms on T4 GPU.

Another resource efficient vision backbone is the Encoder-only Mask Transformer (EoMT) [20]. It achieves detection performance on a par with SOTA models while improving latency. By using only a Vision Transformer [10] (ViT) encoder, EoMT eliminates the need for sequential components like feature adapters, pixel decoders, and separate transformer decoders (*e.g.* Mask2Former [6]). It integrates segmentation directly into the encoder's attention mechanism, enabling parallel processing of image patch and segmentation query tokens in a single forward pass. This approach achieves up to $4\times$ faster inference than Mask2Former, especially with large models (*e.g.* ViT-L), while maintaining competitive accuracy. EoMT's efficiency is further enhanced by large-scale self-supervised pretraining (*e.g.* DINO [3, 36, 42] or EVA-02 [11]), which provide the robust visual representations needed for segmentation without auxiliary components. Its minimalist design makes it ideal for low-latency, real-time applications and easy to integrate into an existing codebase. We therefore choose EoMT as the backbone for DSFlash.

2.5. Evaluation Issues

For a comparison with prior work on PSGG, we use the PSG dataset [51] to report performance results. Like most scene graph datasets, PSG contains no negative ground truth annotations that indicate when a certain relation is predicted incorrectly. Therefore, we have to fall back to ranking metrics like Recall@k ($R@k$) to evaluate model performance. There are some exceptions that address negative ground truth, however these datasets are either only designed for evaluation [29] or based on simulated data only [32]. A well understood problem of using $R@k$ for SGG is the long tail distribution of the used dataset [4, 24–26, 54, 56]. A model that gets only the top 5% of all predicate classes right would already achieve a $R@k$ of 52%. For a more balanced metric, we report Mean Recall@k ($mR@k$) instead, which first computes the $R@k$ per predicate class using all images and averages the individual scores for the final metric.

When evaluating PSGG (and SGG) methods, special care is required when interpreting the model output from various models. Recent work [31, 34] has investigated irregularities in the evaluation protocol of prior SOTA methods. Models generated multiple relation predictions for the same associated ground truth subject-object pair. Consequently, SOTA models artificially increased the number of attempts per ground truth, which is not allowed per *mR@50* definition.

3. Method

We use DSFormer [31] as the baseline for our efficient architecture and provide an overview over the relevant parts in the supplementary. We choose this model as our baseline because of its SOTA performance and simple design, allowing us to quickly iterate new optimizations into the code. Starting from DSFormer, we re-create several parts of the model and end up with a new architecture that is capable to run PSGG at very high speeds while producing more accurate relation predictions.

An overview of the final DSFlash architecture is shown in Fig. 2. To process a single image, DSFlash first uses a pretrained EoMT [20] backbone to extract feature patches and predict a comprehensive set of segmentation masks. For every combination of two segmentation masks (*e.g.* a person

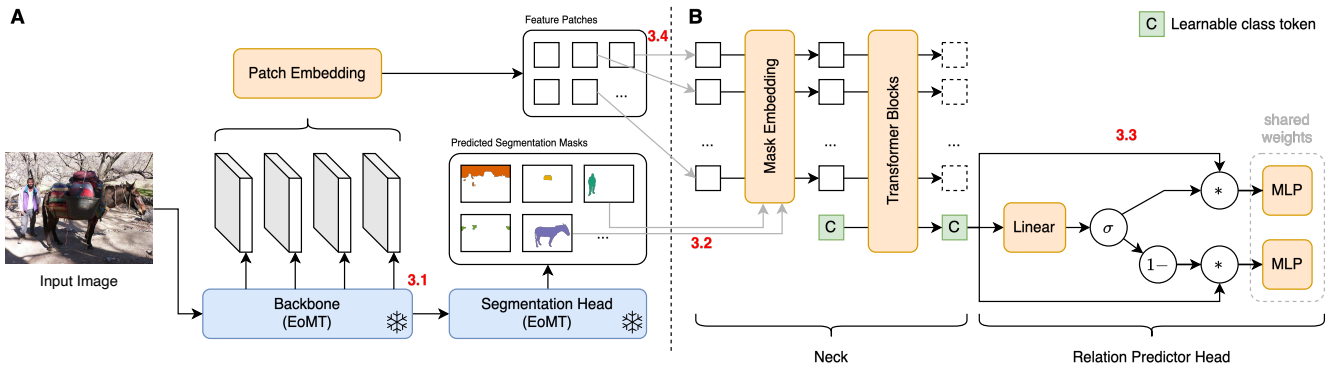
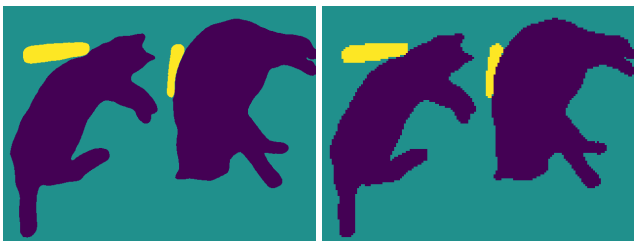


Figure 2. Overview of the DSFlash architecture for inference. Part A is executed once per image. Part B is executed for each combination of two segmentation masks. We use EoMT [20] as the segmentation backbone which is kept frozen throughout the whole training. We use the mask embedding module from DSFormer [31]. The relation predictor head is described in Sec. 3.3. The red numbers indicate which components are covered in which section.

and a donkey), a mask embedding [31] is added to the feature patches from the backbone, encoding the location of subject and object. The enriched patches are then fed through a set of tranformer blocks (model neck) and finally processed by a relation head, predicting the associated relations for the two masks in both directions simultaneously, *e.g.* "person behind donkey" and "donkey right of person".

Note that during training, DSFlash receives ground truth segmentation masks instead of inferred segmentation masks. This provides better supervision through better segmentation masks and reduces training time by skipping the segmentation head.



(a) Upsampled mask logits (b) No upsampling

Figure 3. Qualitative comparison of the segmentation masks produced by EoMT with and without upsampling the logits.

3.1. Merged Backbones

Starting off the architecture of DSFormer, we initially inherit its strictly decoupled approach, meaning that segmentation is delegated to a separate upstream model. Since DSFormer requires the inferred segmentation masks as input, an additional forward pass through a segmentation model is required per frame. Still, DSFormer contains its own ResNet-based backbone which results in costly forward passes through two backbones.

To tackle this inefficiency, we decide to directly extract the feature tensors from the segmentation model, which already runs to produce the segmentation masks. In addition, we replace the slow MaskDINO [23] segmentation model with EoMT [20] which produces panoptic segmentation masks of similar quality at a much lower latency. To get a spatial feature tensor from the backbone, we extract tokens after blocks 2, 5, 8, and 11 for the S and B variants and blocks 5, 11, 17, 23 for the L variant. Since EoMT uses DINO [36, 42], we make sure to only extract patch tokens and exclude the class and register tokens. After block 9 (large: 20), EoMT introduces additional query tokens which we also exclude. Concatenating the extracted tokens along the embedding dimension gives us a feature tensor with the shape $768 \times 40 \times 40$ for a 640×640 input image. Fig. 2 illustrates the approach.

We keep the EoMT-based backbone *frozen at all times*. This speeds up training and enables us to fully outsource panoptic segmentation training which can be performed much more efficiently and on larger datasets without the scene graph generation context.

3.2. Raw-resolution Segmentation Masks

To optimize computational efficiency in the later stages of the architecture, we adopt a ViT-style patch embedding [10], splitting the 40×40 feature tensor into 13×13 patch tokens, each with an embedding dimension of 384. Building on DSFormer, DSFlash integrates the extracted segmentation masks directly into these patch tokens. Specifically, a weighted embedding is added to each patch, where the weight is determined by the fraction of the patch's area that is covered by the corresponding segmentation mask (as formalized in Eq. (1)). A more in-depth description can be found in the supplementary and the original paper [31].

Following DSFormer, we would use the segmentation mask logits from EoMT (160×160), upsample them to the

image size and then compute the patch area overlap fractions. However, since only a resolution of 13×13 is required to compute the embedding, there is no need to upsample to the image size, and we can skip the costly bilinear interpolation step. Fig. 3 visualizes the difference between the different resolutions.

3.3. Bidirectional Predictions

To detect potential relations between two instances in the scene represented by their respective segmentation masks S_0 and S_1, the original DSFormer has to perform two forward passes through the model neck: one to classify the relation with S_0 as the subject and S_1 as the object and another one for the opposite. To reduce the computational cost during inference, we design DSFlash such that it encodes both directions in one single forward pass. This allows us to half the number of predictions required to build a comprehensive scene graph, while keeping the number of parameters in the model almost the same, as we will show in the following. A schematic can be found in Fig. 2 (relation predictor head) and Fig. 4.

Let's assume that the backbone has extracted two segmentation masks S_0 and S_1 for a given image with feature tensor $\mathcal{F}$. The task is now to classify two potential relations. We call the one where S_0 is the subject and S_1 is the object the *forward* prediction $z^{\rightarrow} \in \mathbb{R}^C$ and the reversed one the *backward* prediction $z^{\leftarrow} \in \mathbb{R}^C$, with C being the number of predicate classes. Accordingly, $y^{\rightarrow}, y^{\leftarrow} \in \{0,1\}^C$ are the associated target ground truth values.

Both DSFormer and DSFlash first encode the masks into the image feature tensor $\mathcal{F}$ using a specialized mask embedding module [31], resulting in an enriched feature tensor $x = embed(\mathcal{F}, S_0, S_1) \in \mathbb{R}^D$, with embedding dimension D. DSFormer, then processes x in its prediction head to produce $z^{\rightarrow}$. For $z^{\leftarrow}$, DSFormer computes a second forward pass with $x' = embed(\mathcal{F}, S_1, S_0)$ since $embed$ is not symmetrical.

DSFlash on the other hand uses x for both directions and produces $z^{\rightarrow}$ and $z^{\leftarrow}$ in a single forward pass. Before processing x with a relation predictor head, we disentangle x into two intermediate tensors $t^{\rightarrow}$ and $t^{\leftarrow}$ using a gate mechanism, described in Eqs. (2) to (4). Finally a shared MLP predicts the relations (Eqs. (5) and (6)).

$$g = \sigma\left(gate_{mlp}(x)\right) \in \mathbb{R}^D \tag{2}$$

$$t^{\rightarrow} = g \odot x \in \mathbb{R}^D \tag{3}$$

$$t^{\leftarrow} = (1-g) \odot x \in \mathbb{R}^D \tag{4}$$

$$z^{\rightarrow} = relhead_{mlp}(t^{\rightarrow}) \in \mathbb{R}^C \tag{5}$$

$$z^{\leftarrow} = relhead_{mlp}(t^{\leftarrow}) \in \mathbb{R}^C \tag{6}$$

To learn the relation predictions, we follow DSFormer and use a binary cross entropy loss function between $(z^{\rightarrow}, y^{\rightarrow})$ and $(z^{\leftarrow}, y^{\leftarrow})$.

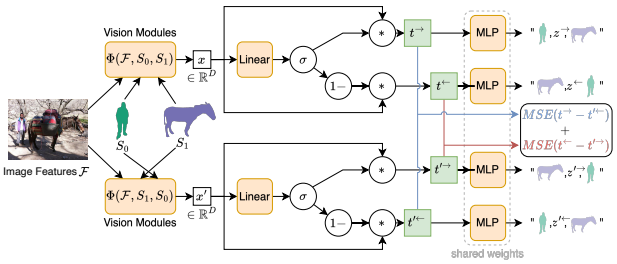


Figure 4. Schematic of DSFlash's gating mechanism and the enforced feature consistency loss during training. Given two segmentation masks and an image, DSFlash computes a class token x using various modules, summarized here as Φ. To train the consistency loss, DSFlash performs two forward passes through the model head with flipped segmentation masks.

In our initial experiments we sorted S_0 and S_1 by their ordering in the ground truth. The model quickly discovered that a positive annotation is 3 times more likely to occur in $y^{\rightarrow}$ than in $y^{\leftarrow}$ in the PSG dataset, making the model rely on a fact that cannot be determined in a real-world application. To prevent this, we introduce the shared relation predictor from Eq. (5) and add an additional feature consistency loss, shown in Eq. (7).

During training, we perform two forward passes through DSFlash's model head for each (S_0, S_1) pair with swapped ordering of the segmentation mask when embedding them into $\mathcal{F}$. This gives us a second enriched feature tensor $x' = embed(\mathcal{F}, S_1, S_0) \neq x$. From here, we compute $t'^{\rightarrow}, t'^{\leftarrow}, z'^{\rightarrow}, z'^{\leftarrow}$ analogously to Eqs. (3) to (6). Ideally, DSFlash would predict $z^{\rightarrow} = z'^{\leftarrow}$ and $z^{\leftarrow} = z'^{\rightarrow}$ because S_0 and S_1 are flipped. To assist the model to learn this similarity, we use an additional consistency loss on the intermediate tensors:

$$\text{Consistency} = \frac{1}{D}\sum_{i=1}^{D}(t_i^{\rightarrow} - t_i'^{\leftarrow})^2 + (t_i^{\leftarrow} - t_i'^{\rightarrow})^2 \tag{7}$$

During inference, DSFlash can rely on its learned ability to treat each directions equally and only a single forward pass is required. More clarifications can be found in Sec. 8 of the appendix.

3.4. Mask-Based Dynamic Patch Pruning

The time spent in the model neck is influenced by the number of processed patch tokens. To reduce the computational cost, we introduce a dynamic patch pruning strategy that is tailored towards DSFlash.

Before the extracted patch tokens from the backbone enter the model neck, an additional subject-object mask embedding is added to each patch token. This mask embedding is directly determined by the overlap ratio of a mask with the respective patch area. Consequently, a patch that is neither

overlapping with the subject nor with the object, receives no additional embedding. Although they provide some context of the overall scene, these patches contain almost no helpful information to determine the final relation predicates. For an additional speedup, we therefore identify the patches that have no overlap with subject or object and drop them before processing them with the model neck. Since the overlaps are computed anyway, the patches to prune can be identified with virtually no computational overhead.

Since the final predictions only rely on information of the classification token, DSFlash can handle variable token numbers.

3.5. Token Merging

We apply token merging [2] to reduce the computational cost when calculating attention in the transformer layers. Specifically, we use ToMe-SD [1, 21] because it unmerges the tokens again which better retains the segmentation capabilities of the backbone. Before every attention layer in DSFlash's backbone, we use ToMe-SD to merge similar tokens. Directly after the attention layer, ToMe-SD unmerges the tokens to the initial number.

3.6. Additional Improvements

To further reduce latency and increase scene graph performance of DSFlash, we perform several improvements.

Efficient subject/object mask encoder. We observe that DSFormer's mask encoder contains inefficient code that involves several copies of the extracted segmentation masks followed by multiple tensor operations that can be exactly represented by an efficient average pooling layer. This is described in more detail in the supplementary.

DeiT III-style augmentation. We choose to follow the data augmentation strategy from DeiT III [47]. We apply random horizontal flips and color jitter, then randomly select one of three transformations: grayscale conversion, solarization, or Gaussian blur.

4. Results

We start with introducing the evaluation metrics and continue with a comparison with other PSGG methods on the PSG dataset. We then analyze various components of DSFlash for the impact on the final performance and speed.

4.1. Metrics

Latency. In Sec. 4, we refer to latency as the average time it takes a PSGG model to process a single image. This evaluates a model's capability to be applied in a real-world environment where image frames have to be processed sequentially from a continuous video stream. To ensure a fair comparison, we only measure the time of a model's forward pass without any data preprocessing steps. All latency values are measured using a batch size of 1 to simulate the single image scenario. We wait for a warmup phase of 200 forward passes before starting the time measurements and report the average latency in the tables and figures.

Table 1. Performance comparison on the PSG dataset [51]. All models are evaluated using a batch size of 1 on an RTX 3090 GPU and the SGDet protocol. Models marked with a star (*) use low-resolution segmentation masks, discussed in Sec. 3.2

Method	mR@50 ↑	Latency ↓	# Params
MotifNet-R50	9.56	100 ms	109M
MotifNet-MD	16.32	504 ms	332M
VCTree-R50	10.14	116 ms	105M
VCTree-MD	17.58	520 ms	327M
HiLo-R50	16.34	277 ms	59M
HiLo-L	19.08	427 ms	230M
REACT	19.00	19 ms	43M
DSFormer	30.70	458 ms	330M
DSFlash-L	**30.90**	50 ms	340M
DSFlash-B*	28.50	23 ms	116M
DSFlash-S*	25.05	**18 ms**	40M

RPS. Since neural networks are highly optimized for modern GPUs, various speed improvements are only noticable if the GPU is working at maximum capcity. Therefore, we also evaluate a model's speed when processing larger batches of data instead of just single-image latency. We report Relations per Frame (*RPS*) as the average number of subject-object pairs that can be processed by the evaluated model per second.

mR@50. To measure scene graph performance, we report *mR@50* [5, 44] which handles the predicate class imbalances in scene graph datasets much better than other metrics like *R@k*. As highlighted in Sec. 2.5, we make sure that multiple mask predictions per ground truth and multiple relation predictions per ground truth are prevented. We evaluate all models using the Scene Graph Detection (SGDet) protocol, sometimes also refered to as SGGen. With this protocol, a PSGG model only receives an image as input and must predict correct segmentation masks, instance labels and subject-predict-object triplets. Additional results with the PredCls protocol are in the supplementary.

4.2. Main Results

A visual comparison of various PSGG methods can be seen in Fig. 1, with more detailed results presented in Tab. 1. DSFlash outperforms all other methods on *mR@50*, even DSFormer by a small margin of 30.9 vs 30.7. At the same time, the largest DSFlash variant which uses an EoMT-L backbone is still faster than all other models with the exception of REACT. Although DSFlash-L has the most parameters in our evaluation, its simple architecture allows it to run efficiently on optimized hardware. On the other side, DSFlash-S* uses

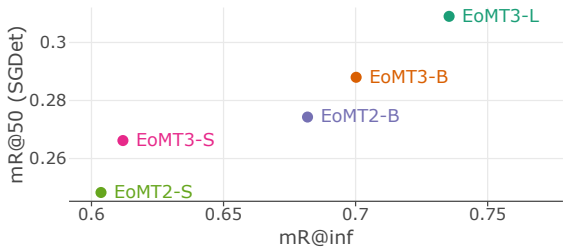


Figure 5. Impact of the segmentation model's capability on the final performance. *mR@inf* [30] is the best possible *mR@k* that a hypothetical perfect PSGG model could achieve, given the extracted segmentation masks from the segmentation model.

an EoMT-S backbone combined with low-resolution segmentation masks, discussed in Sec. 3.1, enabling it to run with a latency of 18 ms on a NVIDIA GeForce RTX 3090 GPU, equivalent to a frame rate of 56 frames per second. With only 40M parameters, it is also the smalles model in our comparison, while still achieving superior *mR@50* scores compared to all other methods, except DSFormer.

We attribute DSFlash's efficiency to its modern backbone with good hardware support, as well as our simple segmentation head that only uses optimized tensor operations.

4.3. Backbone Ablation

As shown in previous literature on two-stage methods, the first segmentation stage has a high impact on the final scene graph performance [31]. We evaluate multiple training runs with equal settings except for the choice of the backbone. As shown in Fig. 5, a larger backbone, is directly correlated to the overall performance. Given the extracted segmentation masks and associated predicted class labels, we report *mR@inf* [30] as the upper limit to *mR@k* for a hypothetical perfect PSGG which has only access to the segmentation masks from the first stage. Fig. 5 shows a striking correlation between *mR@50* and *mR@inf*. In the supplementary, we also show that *mR@50* is directly correlated to the panoptic quality of the segmentation model. As more improved segmentation models are developed in the future, it is very likely that DSFlash can directly leverage their capability for improved scene graph performance.

4.4. DSFlash Improvements

An overview of the most impactful optimizations for DSFlash can be found in Tab. 2. Since we copied many parts from DSFormer, we use it as the baseline model. As expected, the most effective optimization is the use of a single efficient segmentation model instead of two separate slow models, reducing the latency to 41 ms – a 10th of the initial time. At the same time, *mR@50* drops considerably, mostly because DSFlash now works with faster but lower quality segmentation masks form EoMT-3B instead of MaskDINO.

Table 2. Impact of the optimizations on the overall latency, measured on a NVIDIA GeForce RTX 3090 GPU and a batch size of 1. We also report *RPS* as the processed relations per second when processing batched data. The rows are read from top to bottom with each row adding an incremental optimization to the evaluated model. The last row is an exception and is an improvement from the row marked with [1].

Method	mR@50 ↑	Latency (ms) ↓	RPS ↑
Baseline	30.7	445	435
Unified Backbone	25.0	41 (-91%)	5,745
Eff. Mask Embed	25.0	37 (-10%)	7,132
Bidir. Pred.	23.8	29 (-22%)	11,491
Gated Bidir. Pred.[1]	28.8	29 (-0%)	11,491
No Seg. Upscaling	28.5	23 (-21%)	12,928
EoMT-3S	25.1	18 (-22%)	17,897
[1]EoMT-3L	30.9	50 (+72%)	5,996

DSFlash's efficient mask embedding module (Sec. 3.6) reduces latency to 37 ms without affecting *mR@50*. Our new implementation involves much less data copies, mainly resulting in decreased VRAM usage.

Another speed boost is achieved by performing bidirectional predictions (Sec. 3.3), halving the amount of required forward passes through the model neck, which is reflected in the number of processed relations per second on batched data (*RPS*). Since most images from the PSG dataset contain only few objects, the number of forward passes is rarely a bottleneck when measuring single-image latency on a GPU. In addition, our bidirectional training gives the model additional supervision about both relation directions, resulting in an improved *mR@50*.

A very effective technique to reduce latency is to use low-resolution segmentation masks from the segmentation model. This skips a costly bilinear interpolation while providing the same granularity of information to the model neck, as discussed in Sec. 3.1. However, with lower resolution segmentation masks, the final scene graph will also have lower quality and therefore fail to localize some subject and objects from the ground truth. The impact of low-resolution segmentation masks is also dependent on the capability of the backbone, as illustrated in Fig. 6. Compared to DSFlash models with EoMT-B and EoMT-L backbones, the smaller EoMT-S suffers most from lowered segmentation quality. An interesting observation is the fact that a model using EoMT-B with low-resolution segmentation masks runs faster and performs better than a model using EoMT-S with high-resolution segmentation masks. We attribute this to EoMT-S's considerably weaker segmentation capabilities, which directly propagate through the model. It is worth noting that in some scenarios, the EoMT-S-based model might still be preferred since DSFlash with EoMT-S requires only

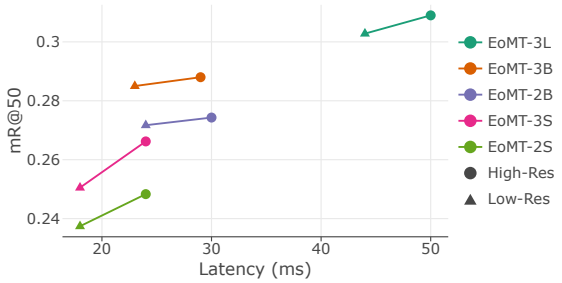


Figure 6. Comparison of the effect of low-resolution segmentation masks on the final performance and latency when using different segmentation backbones.

a third of the parameters that DSFlash with EoMT-B needs.

4.5. Pruning and Merging Methods

We evaluate the impact of dynamic patch pruning (Sec. 3.4) and token merging (Sec. 3.5) on latency and scene graph performance in Tabs. 5 and 6 of the appendix. These techniques complement each other well since they optimize different parts of the DSFlash model. While dynamic patch pruning reduces the number of patch tokens that enter the model neck, we apply token merging to the attention layers of the segmentation backbone.

We evaluate latency using three GPUs: H100, RTX 3090, and GTX 1080. We observe that using the more capable H100 and RTX 3090 GPUs results in no notable latency improvement with a batch size of 1. But when evaluating on batched input, *RPS* still increases. This is most likely because the GPUs are highly optimized for parallel processing and can process all available tokens simultaneously even before pruning. Lower-end GPUs can process less tokens simultaneously. Therefore, reducing the parallel workload via pruning has even a measurable impact on reduced latency with batch size 1.

When using dynamic patch pruning (Sec. 3.4), we notice a slight performance decrease in *mR@50* of 2.13. Even though we have not trained it with pruned patches, DSFlash still manages to handle the altered input to the model neck. For additional results where we have trained DSFlash with dynamic patch pruning, refer to the appendix. The most notable latency improvement can be observed for a 1080 GPU, where latency drops from 230 ms to 205 ms.

A very promising optimization that reduces latency without deteriorating scene graph performance too much is the combination of token merging and dynamic patch pruning. Since both optimization methods affect different parts of the architecture, their latency improvements add up. On a GTX 1080, latency drops from 230 ms to 173 ms while *mR@50* is very similar to our experiment with only patch pruning applied. Using a token merging ratio of 30% combined with patch pruning, gives a similar latency to 50% token merging without patch pruning, while achieving much better *mR@50*.

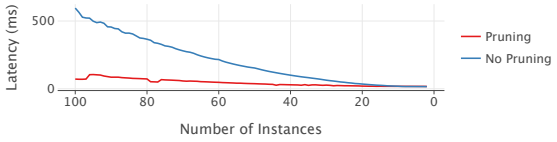


Figure 7. Required time to process an image depending on the number of instances in the scene. The latency was measured on an RTX 3090. With an increasing number of instances, patch pruning becomes more effective.

Patch pruning is especially helpful when working with dense scenes. In dense scenes, instances are often small. Hence, more patches can be dropped, resulting in higher latency improvements, as shown in Fig. 7.

5. Conclusion

In this work, we have introduced DSFlash, a panoptic scene graph generation model that is capable to run with a latency of down to 18 ms on an RTX 3090 GPU while still producing high quality scene graphs.

We have introduced several architectural optimizations and analyze their effect on scene graph quality and latency. While some techniques like mask-based dynamic patch pruning make direct use of DSFlash's architectural design to reduce latency with minimal overhead, we also introduce several methods that can be easily applied to future methods. For example, bidirectional relations enable a scene graph model to predict a scene graph while halving the amount of required forward passes, which directly relates to model throughput on batched inputs. We also experiment with token merging which can be applied to virtually any transformer-based model and show that token merging can be effectively used on resource constrained GPUs.

With the increasing popularity of VLMs in recent research, many systems have grown complex and heavy, often denying the ability to run them on-premise which involves privacy concerns in some areas. While many applications require the complex reasoning and fundamental scene understanding capabilities, other areas might also be approachable using scene graphs as an efficient intermediate step. Combined with DSFlash's near instantaneous high quality scene graphs, we hope that future research further explores ressource efficient systems that can be used as an alternative to the influx of overly complex models.

6. Acknowledgements

The authors gratefully acknowledge the resources on the LiCCA HPC cluster of the University of Augsburg, co-funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 499211671.

References

[1] Daniel Bolya and Judy Hoffman. Token merging for fast stable diffusion. *CVPR Workshop on Efficient Deep Learning for Computer Vision*, 2023. 6

[2] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *International Conference on Learning Representations*, 2023. 6

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 3

[4] Lianggangxu Chen, Youqi Song, Yiqing Cai, Jiale Lu, Yang Li, Yuan Xie, Changbo Wang, and Gaoqi He. Multi-prototype space learning for commonsense-based scene graph generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(2):1129–1137, 2024. 3

[5] Tianshui Chen, Weihao Yu, Riquan Chen, and Liang Lin. Knowledge-embedded routing network for scene graph generation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6156–6164, 2019. 6

[6] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. *arXiv*, 2021. 3

[7] Vinay Damodaran, Sharanya Chakravarthy, Akshay Kumar, Anjana Umapathy, Teruko Mitamura, Yuta Nakashima, Noa Garcia, and Chenhui Chu. Understanding the role of scene graphs in visual question answering, 2021. 1

[8] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024. 3

[9] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 3

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 3, 4

[11] Yuxin Fang, Quan Sun, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. Eva-02: A visual representation for neon genesis. *Image and Vision Computing*, page 105171, 2024. 3

[12] Samir Yitzhak Gadre, Kiana Ehsani, Shuran Song, and Roozbeh Mottaghi. Continuous scene representations for embodied ai. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14829–14839, 2022. 1

[13] Ross Girshick. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015. 2

[14] Rui He, Zehua Fu, Qingjie Liu, Yunhong Wang, and Xunxun Chen. Learning group interaction for sports video understanding from a perspective of athlete. *Frontiers of Computer Science*, 18(4):184705, 2023. 1

[15] Felix Holm, Gözde Ünver, Ghazal Ghazaei, and Nassir Navab. Cat-sg: A large dynamic scene graph dataset for fine-grained understanding of cataract surgery, 2025. 1

[16] Drew A. Hudson and Christopher D. Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[17] Glenn Jocher and Jing Qiu. Ultralytics yolo11, 2024. 3

[18] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023. 2

[19] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Image retrieval using scene graphs. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3668–3678, 2015. 1

[20] Tommie Kerssies, Niccolò Cavagnero, Alexander Hermans, Narges Norouzi, Giuseppe Averta, Bastian Leibe, Gijs Dubbelman, and Daan de Geus. Your ViT is Secretly an Image Segmentation Model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 3, 4

[21] Daniel Kienzle, Marco Kantonis, Robin Schön, and Rainer Lienhart. Segformer++: Efficient token-merging strategies for high-resolution semantic segmentation. *IEEE International Conference on Multimedia Information Processing and Retrieval (MIPR)*, 2024. 6

[22] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73, 2017. 2

[23] Feng Li, Hao Zhang, Huaizhe Xu, Shilong Liu, Lei Zhang, Lionel M. Ni, and Heung-Yeung Shum. Mask DINO: Towards a unified transformer-based framework for object detection and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3041–3050, 2023. 4

[24] Li Li, Wei Ji, Yiming Wu, Mengze Li, You Qin, Lina Wei, and Roger Zimmermann. Panoptic scene graph generation with semantics-prototype learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(4):3145–3153, 2024. 3

[25] Li Li, You Qin, Wei Ji, Yuxiao Zhou, and Roger Zimmermann. Domain-wise invariant learning for panoptic scene graph generation. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3165–3169, 2024.

[26] Nanhao Liang, Yong Liu, Wenfang Sun, Yingwei Xia, and Fan Wang. Ckt-rcm: Clip-based knowledge transfer and relational context mining for unbiased panoptic scene graph generation. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3570–3574, 2024. 3

[27] Weixin Liang, Yanhao Jiang, and Zixuan Liu. GraghVQA: Language-guided graph neural networks for graph-based visual question answering. In *Proceedings of the Third Workshop on Multimodal Artificial Intelligence*, pages 79–86, Mexico City, Mexico, 2021. Association for Computational Linguistics. 1

[28] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. 2

[29] Julian Lorenz, Florian Barthel, Daniel Kienzle, and Rainer Lienhart. Haystack: A panoptic scene graph dataset to evaluate rare predicate classes. In *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2023. 3

[30] Julian Lorenz, Robin Schön, Katja Ludwig, and Rainer Lienhart. A review and efficient implementation of scene graph generation metrics. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2567–2575, 2024. 7

[31] Julian Lorenz, Alexander Pest, Daniel Kienzle, Katja Ludwig, and Rainer Lienhart. A fair ranking and new model for panoptic scene graph generation. In *Computer Vision – ECCV 2024*, pages 148–164, Cham, 2025. Springer Nature Switzerland. 2, 3, 4, 5, 7

[32] Julian Lorenz, Mrunmai Phatak, Robin Schön, Katja Ludwig, Nico Hörmann, Annemarie Friedrich, and Rainer Lienhart. Copa-sg: Dense scene graphs with parametric and proto-relations, 2025. 3

[33] Maëlic Neau, Paulo Santos, Anne-Gwenn Bosser, and Cédric Buche. In defense of scene graph generation for human-robot open-ended interaction in service robotics. In *RoboCup 2023: Robot World Cup XXVI*, pages 299–310, Cham, 2024. Springer Nature Switzerland. 1

[34] Maëlic Neau, Paulo E. Santos, Anne-Gwenn Bosser, Cédric Buche, and Akihiro Sugimoto. React: Real-time efficiency and accuracy compromise for tradeoffs in scene graph generation, 2025. 1, 2, 3

[35] Kien Nguyen, Subarna Tripathi, Bang Du, Tanaya Guha, and Truong Q. Nguyen. In defense of scene graphs for image captioning. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1387–1396, 2021. 1

[36] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision, 2023. 3, 4

[37] Amir M. Rahimi, Kevin Lee, Amit Agarwal, Hyukseong Kwon, and Rajan Bhattacharyya. Toward improving the visual characterization of sport activities with abstracted scene graphs. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4495–4502, 2021. 1

[38] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[39] Isaac Robinson, Peter Robicheaux, Matvei Popov, Deva Ramanan, and Neehar Peri. Rf-detr. `https://github.com/roboflow/rf-detr`, 2025. SOTA Real-Time Object Detection Model. 3

[40] Shaibal Saha and Lanyu Xu. Vision transformers on the edge: A comprehensive survey of model compression and acceleration strategies. *Neurocomputing*, 643:130417, 2025. 3

[41] Jongmin Shin, Enki Cho, Ka Young Kim, Jung Yong Kim, Seong Tae Kim, and Namkee Oh. Towards holistic surgical scene graph, 2025. 1

[42] Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3, 2025. 3, 4

[43] Baoye Song, Shihao Zhao, Zidong Wang, Jianyu Chen, Weibo Liu, and Xiaohui Liu. LW-DETR: a lightweight transformer-based object detection algorithm for efficient railway crossing surveillance. *Complex & Intelligent Systems*, 11(12):480, 2025. 3

[44] Kaihua Tang, Hanwang Zhang, Baoyuan Wu, Wenhan Luo, and Wei Liu. Learning to compose dynamic tree structures for visual contexts. In *Conference on Computer Vision and Pattern Recognition*, 2019. 2, 6

[45] Kaihua Tang, Yulei Niu, Jianqiang Huang, Jiaxin Shi, and Hanwang Zhang. Unbiased scene graph generation from biased training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[46] Yunjie Tian, Qixiang Ye, and David Doermann. Yolov12: Attention-centric real-time object detectors. *arXiv preprint arXiv:2502.12524*, 2025. 3

[47] Hugo Touvron, Matthieu Cord, and Herve Jegou. Deit iii: Revenge of the vit. *arXiv preprint arXiv:2204.07118*, 2022. 6

[48] Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Yolov10: Real-time end-to-end object detection. In *Advances in Neural Information Processing Systems*, pages 107984–108011. Curran Associates, Inc., 2024. 3

[49] Dalin Wang, Daniel Beck, and Trevor Cohn. On the role of scene graphs in image captioning. In *Proceedings of the Beyond Vision and LANguage: inTEgrating Real-world kNowledge (LANTERN)*, pages 29–34, Hong Kong, China, 2019. Association for Computational Linguistics. 1

[50] Tao Wu, Runyu He, Gangshan Wu, and Limin Wang. Sportshhi: A dataset for human-human interaction detection in sports videos. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18537–18546, 2024. 1

[51] Jingkang Yang, Yi Zhe Ang, Zujin Guo, Kaiyang Zhou, Wayne Zhang, and Ziwei Liu. Panoptic scene graph generation. In *ECCV*, 2022. 1, 2, 3, 6

[52] Xuewen Yang, Yingru Liu, and Xin Wang. Reformer: The relational transformer for image captioning. In *Proceedings of the 30th ACM International Conference on Multimedia*, page 5398–5406, New York, NY, USA, 2022. Association for Computing Machinery. 1

[53] Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. Neural motifs: Scene graph parsing with global context. In *Conference on Computer Vision and Pattern Recognition*, 2018. 2

[54] Ao Zhang, Yuan Yao, Qianyu Chen, Wei Ji, Zhiyuan Liu, Maosong Sun, and Tat-Seng Chua. Fine-grained scene graph generation with data transfer. In *ECCV*, 2022. 3

[55] Yishuang Zhao, Qiang Zhang, Xueying Sun, and Guanchen Liu. Integrapsg: Integrating llm guidance with multimodal feature fusion for single-stage panoptic scene graph generation. *Electronics*, 14(17), 2025. 2

[56] Zijian Zhou, Miaojing Shi, and Holger Caesar. HiLo: Exploiting high low frequency relations for unbiased panoptic scene graph generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 21637–21648, 2023. 2, 3

[57] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable {detr}: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*, 2021. 3