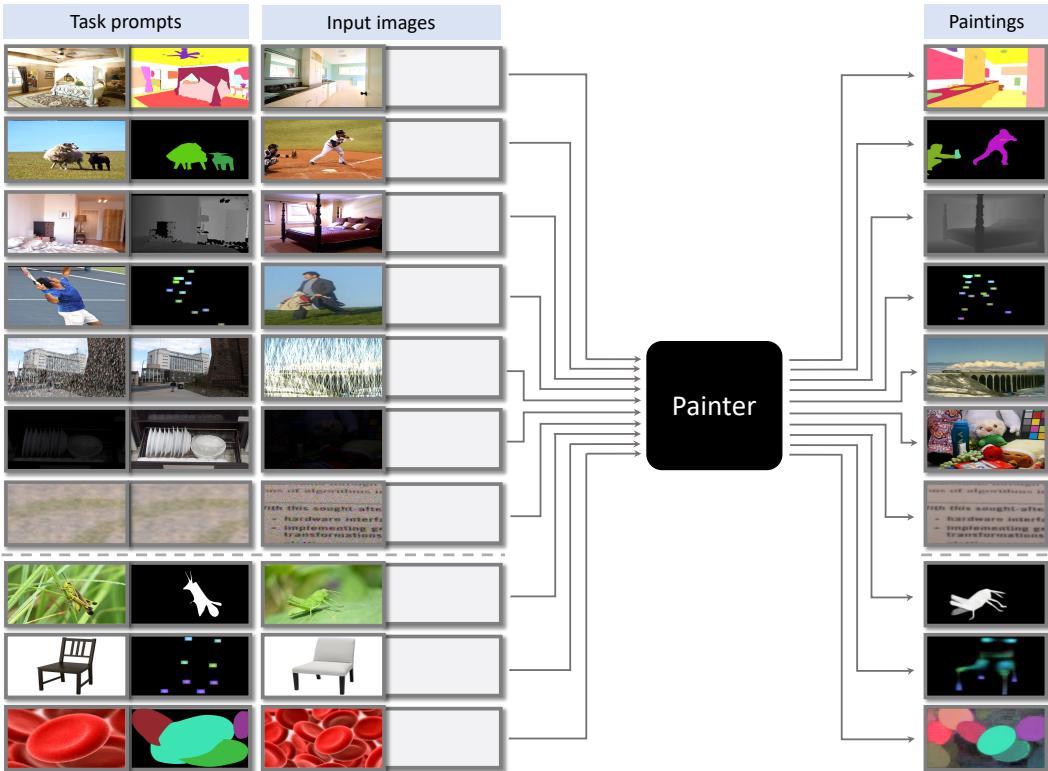


# Images Speak in Images: A Generalist Painter for In-Context Visual Learning

Xinlong Wang<sup>1\*</sup> Wen Wang<sup>2\*</sup> Yue Cao<sup>1\*</sup> Chunhua Shen<sup>2</sup> Tiejun Huang<sup>1,3</sup>

<sup>1</sup> Beijing Academy of Artificial Intelligence    <sup>2</sup> Zhejiang University    <sup>3</sup> Peking University

<https://github.com/baavision/Painter>



**Figure 1.** An illustration of the in-context inference of Painter. Painter is a generalist vision model, which can automatically perform vision tasks according to the input task prompts without the task specific heads. Painter can not only perform in-domain tasks with highly competitive performance, such as semantic segmentation (Row 1), instance segmentation (Row 2), depth estimation (Row 3), keypoint detection (Row 4), denoising (Row 5), deraining (Row 6), and image enhancement (Row7), but also be able to rapidly adapt to various out-of-domain vision tasks using simple prompts, such as open-category object segmentation, keypoint detection, and instance segmentation (Row 8-10).

## Abstract

In-context learning, as a new paradigm in NLP, allows the model to rapidly adapt to various tasks with only a handful of prompts and examples. But in computer vision, the difficulties for in-context learning lie in that tasks vary significantly in the output representations, thus it is unclear how to define the general-purpose task prompts that the vi-

sion model can understand and transfer to out-of-domain tasks. In this work, we present Painter, a generalist model which addresses these obstacles with an “image”-centric solution, that is, to redefine the output of core vision tasks as images, and specify task prompts as also images. With this idea, our training process is extremely simple, which performs standard masked image modeling on the stitch of input and output image pairs. This makes the model capable of performing tasks conditioned on visible image patches. Thus, during inference, we can adopt a pair of input and

\*Equal contribution. Correspondence to [xinlong.wang96@gmail.com](mailto:xinlong.wang96@gmail.com).  
This work is done when Wen Wang is an intern at BAAI.

*output images from the same task as the input condition, to indicate which task to perform. Without bells and whistles, our generalist Painter can achieve competitive performance compared to well-established task-specific models, on seven representative vision tasks ranging from high-level visual understanding to low-level image processing. In addition, Painter significantly outperforms recent generalist models on several challenging tasks.*

## 1. Introduction

Training one generalist model that can execute diverse tasks simultaneously, and even can perform a new task given a prompt and very few examples, is one important step closer to artificial general intelligence. In NLP, the emergence of in-context learning [2, 7, 18] presents a new path in this direction, which uses language sequences as the general interface and allows the model to rapidly adapt to various language-centric tasks with only a handful of prompts and examples.

Thus far, in computer vision, in-context learning is rarely explored and remains unclear how to achieve that. This can be attributed to the differences between two modalities. One difference is that NLP tasks mainly consist of language understanding and generation, so their output spaces can be unified as sequences of discrete language tokens. But vision tasks are the abstractions of raw visual input to varied granularity and angles. Thus, *vision tasks vary significantly in output representations*, leading to various task-specific loss functions and architecture designs. The second difference is that the output space of NLP tasks is even the same as the input. Thus, the task instruction and the example’s input/output, which are all language-token sequences, can be directly used as the input condition (also denoted as the task prompt), which can be processed straightforwardly by the large language model. However, in computer vision, it is unclear how to *define general-purpose task prompts or instructions* that the vision model can understand and transfer to out-of-domain tasks. Several recent attempts [6, 9, 10, 27, 35, 41] tackle these difficulties by following the solutions in NLP. They more-or-less convert vision problems into NLP ones via discretizing the continuous output spaces of vision tasks, and using the language or specially-designed discrete tokens as the task prompts.

However, we believe that images speak in images, *i.e.*, image itself is a natural interface for general-purpose visual perception. In this work, we address the above obstacles with a vision-centric solution. The core observation is that most dense-prediction vision problems can be formulated as *image inpainting*, *i.e.*:

*Given an input image, prediction is to inpaint the desired but missing output “image”.*

Thus, we need a representation of 3-channel tensor that

appears as an “image” for the output of the vision tasks, and specify the task prompts using a pair of images. Here we showcase several representative vision tasks for training, including depth estimation, human keypoint detection, semantic segmentation, instance segmentation, image denoising, image deraining, and image enhancement, and unify their output spaces using a 3-channel tensor, *a.k.a.* “output image”. We carefully design the data format for each task, such as instance mask and per-pixel discrete labels of panoptic segmentation, per-pixel continuous values of depth estimation, and high-precision coordinates of pose estimation. Including more tasks is very straightforward, as we only need to construct new data pairs and add them to the training set, without modifications to either the model architecture or loss function.

Based on this unification, we train a generalist Painter model with an extremely simple training process. During training, we stitch two images from the same task into a larger image, and so do their corresponding output images. Then we apply masked image modeling (MIM) on pixels of the output image, with the input image being the condition. With such a learning process, we enable the model to perform tasks conditioned on visible image patches, that is, the capability of in-context prediction with the visual signal as context.

Thus the trained model is capable of the in-context inference. That is, we directly use the input/output paired images from the same task as the input condition to indicate which task to perform. Examples of in-context inference are illustrated in Figure 1, consisting of seven in-domain examples (seven rows at top) and three out-of-domain examples (three rows at bottom). This definition of task prompts does not require deep understanding of language instructions as need by almost all previous approaches, and makes it very flexible for performing both in-domain and out-of-domain vision tasks.

Without bells and whistles, our model can achieve competitive performance compared to well-established task-specific models, on several fundamental vision tasks across high-level visual understanding to low-level image processing, namely, depth estimation on NYUv2 [37], semantic segmentation on ADE-20K [54], human keypoint detection on COCO [31], panoptic segmentation on COCO [31], and three low-level image restoration tasks. Notably, on depth estimation of NYUv2, our model achieves state-of-the-art performance, outperforming previous best results by large margins which have heavy and specialized designs on architectures and loss functions. Compared to other generalist models, Painter yields significant improvements on several challenging tasks.

## 2. Related Work

**Unified Modeling** The emergence of Transformer [40] provides the possibility to share the basic modeling module across different modalities. Until now, Transformers are widely-adopted in language [7, 12, 32], vision [4, 8, 9, 14, 33], speech [13, 17, 24] and multimodal [23, 34, 38] domains. Perceiver [22] and Perceiver-IO [21] are the first attempts to use the exact same Transformer architecture in different domains, such as natural language and visual understanding processing, StarCraft II, and multi-modal domains. If the input could be transformed to a sequence of tokens, one can adopt Transformer for modeling the relationships between different tokens.

**Vision Generalist** Due to the general modeling capability of Transformer, there are some efforts to unify different tasks in vision domains, resulting in several vision generalists [9, 10, 27, 35, 55]. DETR [8] first adopted Transformer as the task specific head for object detection. Based on this, Pix2Seq [9] defined the output space of object detection as a discrete space, and conduct this task in an auto-regressive manner. Due to the fundamental nature of object detection, Pix2Seq provides a direction for unifying different vision tasks using discrete spaces, thus motivating a lot of following work. Unified-IO [35] and OFA [41] both homogenize the diverse inputs and outputs to a sequence of discrete tokens, perform joint modeling in a sequence-to-sequence manner over vision, vision & language and NLP tasks, and use T5-style architectures [36] with billions of parameters, where Unified-IO unifies more tasks than OFA with larger size of models. Pix2Seq v2 [10] unified object detection, instance segmentation, keypoint estimation and image captioning in the same defined discrete spaces as Pix2Seq. UViM [27] unified pixel-labeling tasks with the same modeling approach but trained separate models for different tasks, such as panoptic segmentation, depth estimation and colorization.

Notably, from our point of view, the input of visual signals is continuous in nature, thus we try to make the output space of several representative vision tasks as continuous as images to reduce the quantization error caused by discretization and further enable the in-context visual learning with masked image modeling.

**In-Context Learning** For the first time, GPT-3 [7] defined a new learning paradigm, in-context learning, where a series of NLP tasks can be formulated as the text completion task given prompts and examples. In-context learning grants models new capabilities to perform on-the-fly computational reasoning or novel-pattern recognition that is unlikely to have occurred in training. Flamingo [2] extended the input of the large language models to not only texts but also images and videos, but still used languages as the general interface, such that the model can perform many visual-linguistic tasks given prompts and examples,

such as image captioning, visual question answering, optical character recognition (OCR), etc. In other domains, it appears non-trivial to directly introduce the in-context learning capability. AD [28] uses algorithm distillation to combine in-context capability with reinforcement learning. In computer vision, a concurrent work [6] performs inpainting on the figures and infographics from vision articles, but only works for predicting on discrete space as the language domain, and shows the in-context capability on foreground segmentation, single object detection and colorization. While the work [6] proves the concept of in-context learning for vision tasks, no results were reported on standard benchmark datasets. Thus, it remains unclear how the method performs on real-world datasets. In contrast, our model works well with masked image modeling on pixels on seven diverse and challenging vision tasks, including depth estimation, keypoint estimation, semantic segmentation, panoptic segmentation, image denoising, image deraining, and image enhancement, and also shows *highly competitive* performance on these tasks.

## 3. Approach

The core idea of our framework is to reformulate most vision tasks such as depth estimation, semantic segmentation, instance segmentation, keypoint detection and image restoration as an image inpainting problem. To do so, we redefine the output space of those tasks as “images”.

### 3.1. Redefining Output Spaces as “Images”

We denote an input image as  $\mathbf{x}$  with the size of  $H \times W \times 3$ , and standard definition of the task ground truth as  $\mathbf{y}^t$  which has various sizes for different task  $t$ , and we redefine these task outputs still in the image space, denoting as  $\hat{\mathbf{y}}^t$  with the size of  $H \times W \times 3$ . Our philosophy is to keep the spatial relationships between pixels to be intact, and each pixel of the output image still represents the output for this task of the corresponding input image pixel but in the RGB space. That is,  $\hat{\mathbf{y}}_{i,j}^t$  with three dimensions denotes the corresponding ground truth of input pixel  $\mathbf{x}_{i,j}$ . We select seven representative vision tasks with diverse types of outputs, such as depth estimation, semantic segmentation, keypoint detection and panoptic segmentation. Here we show how we redefine the per pixel ground-truth for each task as a 3-channel tensor, similar to the RGB space. Note that in theory, a fixed number of output channels can serve our purpose. We choose 3 channels to make an image.

**Monocular depth estimation** is a dense prediction task with weak semantics, to estimate the per-pixel depth value (distance relative to the camera) given an input RGB image. For NYUv2 [37], the per-pixel ground-truth depth  $\mathbf{y}_{i,j}^t$  is a real value in the range of  $[0, 10]$  meters. Here we map the ground-truth value from real-valued range  $[0, 10]$  to the integer space with range  $[0, 255]$ ,  $\hat{\mathbf{y}}_{i,j,0}^t = \lfloor \mathbf{y}_{i,j}^t \times \frac{255}{10} \rfloor$ , and

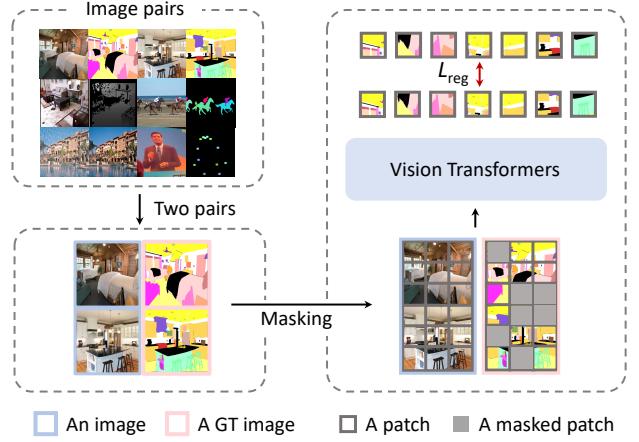
let the three channels  $\hat{y}_{i,j,0}^t$ ,  $\hat{y}_{i,j,1}^t$  and  $\hat{y}_{i,j,2}^t$  be the same ground truth. In inference, we directly average the outputs of the three channels and then perform the inverse linear transformation of the training to obtain a depth estimate in the range of [0, 10].

**Semantic segmentation** is a dense prediction task with strong semantics, to predict the per-pixel semantic label given an input image. Given a semantic segmentation task with  $L$  categories, we let the RGB space to represent these  $L$  categories with the same margin in each space. We represent  $L$  as a 3-digit number with  $b$ -base system, where  $b = \lceil L^{\frac{1}{3}} \rceil$ , and  $\hat{y}_{i,j,0}^t$ ,  $\hat{y}_{i,j,1}^t$  and  $\hat{y}_{i,j,2}^t$  represent their hundreds, tens and ones places, with a margin defined as  $m = \lfloor \frac{256}{b} \rfloor$ . For example, ADE-20K [54] has 150 semantic categories with one background class, thus we set the base as  $b = 6$ , and the margin as  $m = 42$ . The output channels are defined as  $\hat{y}_{i,j,0}^t = \lfloor \frac{l}{b^2} \rfloor \times m$ ,  $\hat{y}_{i,j,1}^t = \lfloor \frac{l}{b} \rfloor \bmod b \times m$ , and  $\hat{y}_{i,j,2}^t = l \bmod b \times m$ , where  $l$  denotes the corresponding category and is an integer value in range of [0,  $L$ ). In inference, we discretize the output of each pixel with the margin  $m$  and obtain its corresponding category.

**Keypoint detection** is a fine-grained localization task to simultaneously detect the objects (*e.g.*, human) and localize their detailed keypoints. We follow recent heatmap-based top-down pipeline [46], thus it is defined as a 17-category point-localization task for human keypoint detection. For each keypoint, we need to localize it to its corresponding pixel, which is very fine-grained. We decouple this task into a combination of 17-category keypoint classification using two channels of  $\hat{y}_{i,j,1}^t$  and  $\hat{y}_{i,j,2}^t$ , and class-agnostic keypoint localization using another channel of  $\hat{y}_{i,j,0}^t$ . For the 17-category classification task, we define each keypoint as a  $9 \times 9$  pixel square, and define the color of each square using the approach in semantic segmentation. For the class-agnostic point localization task, we define 17 squares, each of which is a  $17 \times 17$  heatmap with Gaussian distribution that the center pixel with largest value of 255 is the position of the ground truth keypoint. In inference, we obtain the category and location for each keypoint as the final results from the 3-channel output image.

**Panoptic segmentation** is a combination of semantic segmentation task and instance segmentation task. Thus, we perform these two tasks separately for ease of redefinition of output space and optimization, and then combine their results to obtain the results for panoptic segmentation.

Here we introduce the redefinition of output space of class-agnostic instance segmentation. For this task, we directly change the color of each instance mask in the image to the same one, thus different instances use different colors. In theory, we can randomly choose colors for different instances, but we find that this setting would make the model hard to optimize. To address this optimization issue, we



**Figure 2. The training pipeline of the masked image modeling (MIM) framework.**

follow SOLO [42] to assign the color of each instance mask according to the absolute position of its center in the image. We conceptually divide the image into  $16 \times 20 \times 20$  blocks, corresponding to three channels respectively. We assign a fixed color to each block, then color the mask accordingly if its center locates in that block. In inference, we adopt each color as a kernel to compute the distance with each pixel in the image, and then set a threshold to get the final masks. To get the category for each instance mask, we directly assign the majority category in the semantic segmentation result within each instance mask as its category.

**Image restoration** takes the corrupted image as input and outputs the corresponding clean image. In this paper, we investigate three representative image restoration tasks, including image denoising, image deraining, and low-light image enhancement. Since both the input and output are inherently defined in the RGB space, these tasks can be seamlessly unified in our Painter model without any transformation.

### 3.2. A Masked Image Modeling Framework

Based on the redefinition of the output spaces of above representative vision tasks, the input and output of these tasks are all images. Thus, here we directly apply a standard masked image modeling (MIM) pipeline [5, 19, 48] for training, illustrated in Figure 2. This framework consists of three major components: input format, architecture and loss function.

**Input Format** During training, each input sample is the concatenation of two pairs of images from the same task, which have already been applied data augmentations separately, as shown in the left part of Figure 2. Each image pair consists of one image and its corresponding task output, which is also redefined as an image. We randomly mask the task output image and train the model to reconstruct the missing pixels. For the masked area, we follow the NLP

community [32, 38] and previous works [5, 19, 48] to use a learnable token vector to replace each masked patch. We adopt the block-wise masking strategy and find the masking ratio as 75% to work well.

**Architecture** We adopt a vanilla vision Transformer (ViT) [14] as the encoder, which consists of stacked Transformer blocks, *e.g.*, 24 blocks for ViT-large. We concatenate 4 feature maps evenly sampled from these blocks and use a simple three-layer head to map the features of each patch to its original resolution, *e.g.*,  $16 \times 16 \times 3$ . Specifically, the head consists of a linear ( $1 \times 1$  convolution) layer, a  $3 \times 3$  convolution layer, and another linear layer.

Since each input sample includes both the input image and its output image, the input resolution can be larger than the traditional training process, and also the computation cost. To address this problem, we propose to reduce the computation cost by merging the early features of the input image and the output image. Specifically, we feed the input image and the output image to the model in parallel, then add their features patch by patch after a few blocks, *e.g.*, 3 blocks by default. This design saves nearly half of the computation costs, but we find no degradation in performance.

**Loss Function** We use a simple regression loss to train Painter. Specifically, smooth- $\ell_1$  [16] loss is computed on the masked pixels. We also consider  $\ell_1$  and  $\ell_2$  in our experiments, and we find that the smooth- $\ell_1$  loss achieves the best performance.

### 3.3. In-Context Inference

At the first time, we design an in-context inference procedure, which is very flexible for performing both in-domain and out-of-domain vision tasks. As the input and output spaces of vision tasks have been unified as images, we can directly use the input/output paired images from the same task as the input condition (task prompts) to indicate which task to perform, and concatenate them with the input image and a masked image for completing the corresponding task. Examples are shown in Figure 1. This definition of task prompt does not require deep understanding of language instructions like previous approaches, but uses the visual signal as context which can be understood by the vision model and well matches the nature of the visual domain.

Also, different task prompts would lead to different results. Thus, how to select or generate a more suitable task prompt can be a new direction to explore. Here we present two simple baselines and we leave more explorations as the future work. The first baseline is to obtain a better prompt via selection, that we traverse the whole training set in a heuristic manner and select the best-performing example pair for each task. The second baseline is to generate a task prompt. We define the task prompt as the learnable tensors, freeze the whole model, and then use the training loss to optimize the task prompts. We compare these two solutions

with the random counterpart in the experiments with the visualizations, as shown in §4.3.

## 4. Experiments

### 4.1. Settings

**Datasets and input format** NYUv2 [37] dataset consists of 464 indoor scenes captured by a Microsoft Kinect camera. The official training split (24K images) is used for training, and we report the Root Mean Square Error (RMSE), absolute mean relative error (A.Rel) and the percentage of inside pixels with different thresholds of  $\delta$  on the 654 testing images from 215 indoor scenes.

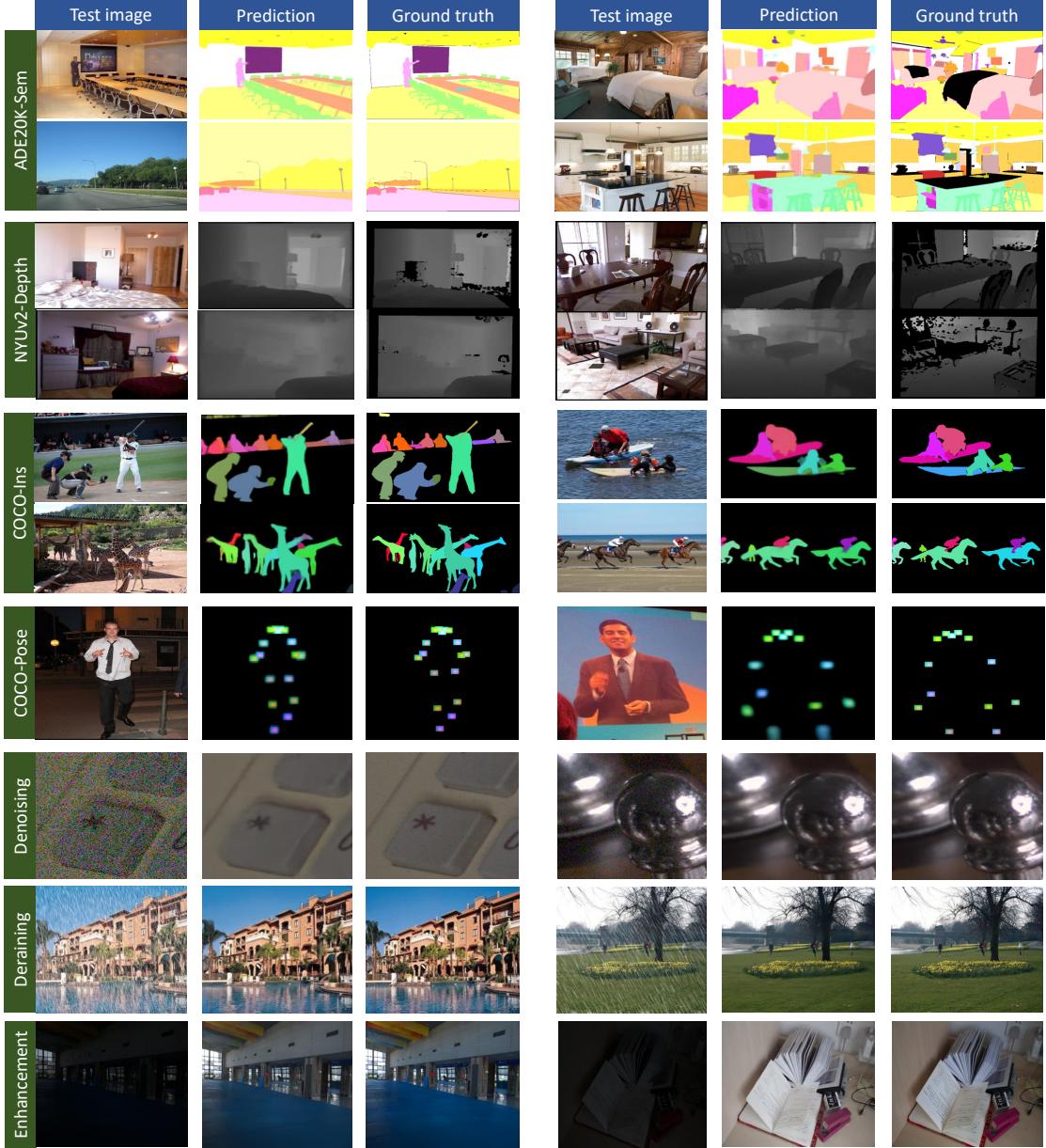
ADE20K [54] is a widely-used semantic segmentation dataset, covering a broad range of 150 semantic categories. It has 25K images in total, with 20K for training, 2K for validation, and another 3K for testing. We adopt the widely-used metric of mean IoU (mIoU) for evaluation.

MS-COCO [31] contains approximately 118K training images and 5K validation images used for evaluation, with 80 “things” and 53 “stuff” categories. Panoptic segmentation task is evaluated on the union of “things” and “stuff” categories. During training, we generate the output images of semantic segmentation with 133 categories, and the class-agnostic instance segmentation with only “things” categories. During inference, we perform inference twice for each input image to obtain the results of semantic segmentation and class-agnostic instance segmentation, respectively, then merge them together to get the results of panoptic segmentation. We report panoptic quality as the measure.

For human keypoint detection, we use the standard person detection results from Simple Baseline [46], which use the standard splits on COCO with 15K training samples and validation set for evaluation, and report the AP based on OKS as the evaluation metric [46].

Image restoration tasks are evaluated on several popular benchmarks, including SIDD [1] for image denoising, LoL [45] for low-light image enhancement, and the merged deraining dataset [51] for deraining. More details about these datasets are provided in Table S1.

**Training details** During training, we employ an AdamW optimizer [25] with a cosine learning rate scheduler, and train for 54K iterations. The training hyper-parameters are: the batch size as 2048, base learning rate as  $1e-3$ , weight decay as 0.05,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ , drop path [20] ratio as 0.1, warm-up for 5.4K iterations. A light data augmentation strategy is used: random resize cropping with scale range of  $[0.3, 1]$  and aspect ratio range of  $[3/4, 4/3]$ , followed by a random flipping. The input image size is  $448 \times 448$  by default. The sampling weight for each task is 0.1 (NYUv2 depth estimation), 0.2 (ADE-20K semantic segmentation), 0.15 (COCO class-agnostic instance segmentation), 0.25 (COCO semantic segmentation), 0.2 (COCO



**Figure 3.** Visualizations of examples and predictions obtained by our Painter for different tasks, such as semantic segmentation, depth estimation, instance segmentation, human keypoint detection, image denoising, image deraining and low-light image enhancement. Best viewed on screen.

human keypoint detection), 0.15 (image denoising), 0.05 image deraining, and 0.05 (low-light image enhancement).

## 4.2. Results

**System-level comparison** With the corresponding task prompts, we compare our approach, Painter, with recent best vision generalist models and specialized models on seven representative tasks, shown in Table 1. Without task-specific design, our Painter sets new records on NYUv2 depth estimation, outperforming not only vision generalists, such as Unified-IO [35] and UVIM [27], but also the state-of-the-art

specialized model, *e.g.*, BinsFormer [30]. For COCO keypoint detection, Painter significantly outperforms the generalist model Pix2Seq v2 [10] by 7.3 AP. For ADE-20K semantic segmentation, COCO panoptic segmentation, and the three low-level image processing tasks, our model achieves comparable performance to those well-designed task-specific models. There is still much room for boosting our approach compared to other well-designed specialized models. For example, our default input image size is  $448 \times 448$  while those specialized panoptic segmentation models use a much larger resolution, *e.g.*,  $1024 \times 1024$  in Mask2Former [11].

	depth estimation			semantic seg.	panoptic seg.	keypoint det.	denoising		deraining		enhance.	
	NYUv2			ADE-20K	COCO	COCO	SIDD					LoL
	RMSE ↓	A.Rel ↓	$\delta_1 \uparrow$	mIoU ↑	PQ ↑	AP ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑
specialized models												
DenseDepth [3]	0.465	0.123	0.846	-	-	-	-	-	-	-	-	-
BinsFormer [30]	0.330	0.094	0.925	-	-	-	-	-	-	-	-	-
UperNet-ViT-L [47]	-	-	-	49.9	-	-	-	-	-	-	-	-
Mask2Former [11]	-	-	-	57.7	57.8	-	-	-	-	-	-	-
DETR [8]	-	-	-	-	45.6	-	-	-	-	-	-	-
HRNet [46]	-	-	-	-	-	76.3	-	-	-	-	-	-
HRFFormer [11]	-	-	-	-	-	77.2	-	-	-	-	-	-
Uformer [44]	-	-	-	-	-	-	39.89	0.960	-	-	-	-
MPRNet [50]	-	-	-	-	-	-	39.71	0.958	32.73	0.921	-	-
MIRNet-v2 [51]	-	-	-	-	-	-	39.84	0.959	-	-	24.74	0.851
generalist framework, specialized models												
UViM [27]	0.467	-	-	-	45.8	-	-	-	-	-	-	-
generalist models												
Unified-IO [35]	0.385	-	-	-	-	-	-	-	-	-	-	-
Pix2Seq v2 [10]	-	-	-	-	-	64.8	-	-	-	-	-	-
Painter (ours)	0.288	0.080	0.950	49.9	43.4	72.1	38.88	0.954	29.49	0.868	22.40	0.872

**Table 1.** System-level comparison with the vision generalist models, and the recent best specialized models on seven representative tasks covering high-level visual understanding and low-level image processing. We compare with the best results of each method. The backbones of the listed generalist methods are: ViT-L for UViM, Unified-IO<sub>XL</sub> with 2925M parameters, ViT-B with another Transformer decoder for Pix2Seq v2, and ViT-L for Painter.

	depth estimation			semantic seg.	panoptic seg.	keypoint det.	denoising		deraining		enhance.	
	NYUv2			ADE-20K	COCO	COCO	SIDD					LoL
	RMSE ↓	A.Rel ↓	$\delta_1 \uparrow$	mIoU ↑	PQ ↑	AP ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑
separate training	0.327	0.090	0.930	47.2	41.3	72.5	38.43	0.956	28.28	0.844	21.67	0.850
joint training	0.288	0.080	0.950	49.9	43.4	72.1	38.88	0.954	29.49	0.868	22.40	0.872

**Table 2.** Comparison of the models with the settings of joint training vs. separate training on seven representative tasks. For the separate training setting, we train the model separately on each task with the same number of iterations as joint training on this task.

Achieving state-of-the-art performance on every task is not the major goal of this paper, and we leave this as future work.

**Joint training vs. separate training** We compare two training settings, *i.e.*, joint training and separate training, in Table 2. For the separate training setting, we train each model separately on each task with the same number of iterations and architectures as joint training on this task. We can observe that models with joint training generally outperform that with separate training on most of the tasks, indicating that our in-context training approach with the unification of output spaces can somewhat benefit them from each other. But conflict may still exist, *e.g.*, joint training performs slightly worse on keypoint detection. Exploring the relationships between tasks in our simple and unified framework could be an interesting and promising direction.

**Qualitative results** To demonstrate the capability of our generalist model in an intuitive perspective, we visualize the task output of the selected images from the validation set of several in-domain tasks, such as semantic segmentation,

depth estimation, instance segmentation, human keypoint detection, image denoising, image deraining, and low-light image enhancement. As shown in Figure 3, Painter can make very accurate predictions on all these tasks.

### 4.3. Prompt Tuning

Different task prompts would lead to varied results. Here we compare three simple baselines, ‘random’, ‘searched’ and ‘learned’ prompts. Random prompts denote to randomly select one example from that task as the prompt. Searched prompt denotes to traverse the training set in a heuristic manner and select the best-performing example pair for each task. Learned prompt denotes that we define the task prompt as the learnable tensors, freeze the whole model, and then use the training loss to optimize the task prompts.

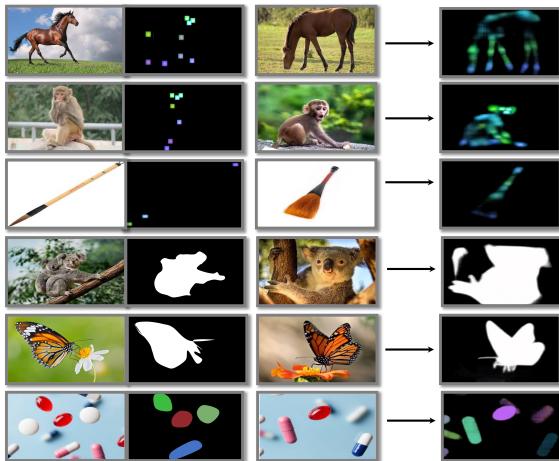
The qualitative comparisons are shown in Table 3. We can see that the model using the searched and learned prompts perform better than that with random ones, which indicates that the optimization process helps the task to find a more suitable prompt. Also, the model with random prompts

	depth estimation			semantic seg.		panoptic seg.		keypoint det.		denoising		deraining		enhance.	
	NYUv2			ADE-20K		COCO		COCO		SIDD		5 datasets		LoL	
	RMSE ↓	A.Rel ↓	$\delta_1 \uparrow$	mIoU ↑	PQ ↑	AP ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑	
random prompts	0.291	0.080	0.949	49.3	43.1	71.8	38.46	0.953	29.26	0.865	22.31	0.871			
searched prompts	0.288	0.080	0.950	49.9	43.4	72.1	38.88	0.954	29.49	0.868	22.40	0.872			
learned prompts	0.286	0.080	0.949	49.9	43.3	72.2	38.71	0.954	29.56	0.870	22.38	0.872			

**Table 3.** Comparison of the models with random prompts, searched prompts, and learned prompts for different tasks.

Model	mIoU ↑
MAE-VQGAN [6]	58.3
Painter	62.3

**Table 4.** Quantitative results on open-vocabulary FSS-1000.



**Figure 4.** Visualizations of examples and predictions obtained by Painter, such as open-vocabulary keypoint detection (e.g., horse, monkey and broom), object segmentation (koala and butterfly) and instance segmentation (tablets).

works relatively well, indicating the robustness of our model. In Figure S6, we show the visualizations of the learned prompt images for different tasks.

It is very promising that the performance of a task can be improved by optimizing the input prompt. This provides the possibility that if a category or task does not appear in the training data, we can still achieve relatively good performance on this task by optimizing the prompts, without tuning the parameters of the model, which is also one core advantage of in-context learning.

#### 4.4. Generalization

The core property of in-context learning is that it allows the model to rapidly adapt to various tasks with only a handful of prompts and examples. Here, we explore this capability via visualizations, shown in Figure 1, Figure 4, and Figure S5. From these visualizations, we find that our model can perform the task seen in training but with input im-

ages that their categories are unseen in training, such as open-vocabulary keypoint detection (*e.g.*, monkey, horse and broom), object segmentation (koala) and instance segmentation (tablets). In addition, we quantitatively evaluate Painter on a few-shot segmentation benchmark which requires to segment objects of 1k novel classes. Painter largely outperforms a concurrent work [6], as reported in Table 4,

## 5. Discussion and Conclusion

In this work, for the first time, we explore how to perform in-context visual learning and present a vision-centric solution in which the context is defined as visual signals. Thus, the models are granted with the capability to perform tasks conditioned on the paired data from that task, that is, in-context inference. Our approach achieves highly competitive performance on a representative and diverse set of seven tasks.

This work is not without drawbacks. First, there is still much room for boosting our approach especially on the difficult task of panoptic segmentation, comparing to the specialized models. In addition, since our approach is designed based on visual signals as contexts, this general interface does not seem natural for modeling language signals. How to model discrete language signals as continuous ones seems to be an impressive direction, and some work has started to emerge recently.

While there are previous approaches that hope to use general interfaces to solve multiple vision tasks, we may be the first to grant models the ability to learn and complete tasks in context, which does have the opportunity to handle out-of-domain tasks. We hope this work will draw attention to this promising direction, and we believe that the best GPT-3 moment in the vision field is yet to come.

## Acknowledgement

This project is supported by the National Key R&D Program of China (2020AAA0105200). We would like to thank Hanxiao Qu, Yemin Shi, Yan Tian, and Xigang Cao for their help on the GPU resources, Xinxin Liu and Kai Lu for beautifying the figures, as well as other colleagues at Beijing Academy of Artificial Intelligence for support throughout this project.

## References

- [1] Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown. A high-quality denoising dataset for smartphone cameras. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.
- [2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. *Adv. Neural Inform. Process. Syst.*, 2022.
- [3] Ibraheem Alhashim and Peter Wonka. High quality monocular depth estimation via transfer learning. *arXiv preprint arXiv:1812.11941*, 2018.
- [4] Roman Bachmann, David Mizrahi, Andrei Atanov, and Amir Zamir. Multimae: Multi-modal multi-task masked autoencoders. *arXiv preprint arXiv:2204.01678*, 2022.
- [5] Hangbo Bao, Li Dong, and Furu Wei. BEiT: BERT pre-training of image Transformers. *arXiv preprint arXiv:2106.08254*, 2021.
- [6] Amir Bar, Yossi Gandelsman, Trevor Darrell, Amir Globerson, and Alexei A. Efros. Visual prompting via image inpainting. *Adv. Neural Inform. Process. Syst.*, pages 1–24, 2022.
- [7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.
- [8] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Eur. Conf. Comput. Vis.*, pages 213–229, 2020.
- [9] Ting Chen, Saurabh Saxena, Lala Li, David J. Fleet, and Geoffrey Hinton. Pix2seq: A language modeling framework for object detection. *Int. Conf. Learn. Representations*, pages 1–17, 2021.
- [10] Ting Chen, Saurabh Saxena, Lala Li, Tsung-Yi Lin, David J. Fleet, and Geoffrey Hinton. A unified sequence interface for vision tasks. *Adv. Neural Inform. Process. Syst.*, 2022.
- [11] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. *arXiv preprint arXiv:2112.01527*, 2021.
- [12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *Annual Conf. North American Chapter of the Association for Computational Linguistics*, 2018.
- [13] Linhao Dong, Shuang Xu, and Bo Xu. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *IEEE Int. Conf. Acoustics, Speech & Signal Process.*, pages 5884–5888, 2018.
- [14] 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 *Int. Conf. Learn. Representations*, 2021.
- [15] Xueyang Fu, Jiabin Huang, Delu Zeng, Yue Huang, Xinghao Ding, and John Paisley. Removing rain from single images via a deep detail network. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.
- [16] Ross Girshick. Fast R-CNN. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2015.
- [17] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.
- [18] Yaru Hao, Haoyu Song, Li Dong, Shaohan Huang, Zewen Chi, Wenhui Wang, Shuming Ma, and Furu Wei. Language models are general-purpose interfaces. *arXiv preprint arXiv:2206.06336*, pages 1–32, 2022.
- [19] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021.
- [20] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger. Deep networks with stochastic depth. In *Eur. Conf. Comput. Vis.*, pages 646–661, 2016.
- [21] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Kopputla, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver IO: A general architecture for structured inputs and outputs. *Int. Conf. Learn. Representations*, 2021.
- [22] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *Int. Conf. Machine Learn.*, pages 4651–4664. PMLR, 2021.
- [23] Aishwarya Kamath, Mannat Singh, Yann LeCun, Gabriel Synnaeve, Ishan Misra, and Nicolas Carion. Mdetr-modulated detection for end-to-end multi-modal understanding. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1780–1790, 2021.
- [24] Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki, Nelson Enrique Yalta Soplin, Ryuichi Yamamoto, Xiaofei Wang, et al. A comparative study on transformer vs rnn in speech applications. In *IEEE Automatic Speech Recognition and Understanding Workshop*, pages 449–456, 2019.
- [25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [26] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6399–6408, 2019.
- [27] Alexander Kolesnikov, André Susano Pinto, Lucas Beyer, Xiaohua Zhai, Jeremiah Harmsen, and Neil Houlsby. UViM: A unified modeling approach for vision with learned guiding codes. *Adv. Neural Inform. Process. Syst.*, 2022.
- [28] Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven

- Hansen, Angelos Filos, Ethan Brooks, Maxime Gazeau, Himsnu Sahni, Satinder Singh, and Volodymyr Mnih. In-context reinforcement learning with algorithm distillation. *Adv. Neural Inform. Process. Syst.*, pages 1–21, 2022.
- [29] Yu Li, Robby T Tan, Xiaojie Guo, Jiangbo Lu, and Michael S Brown. Rain streak removal using layer priors. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.
- [30] Zhenyu Li, Xuyang Wang, Xianming Liu, and Junjun Jiang. BinsFormer: Revisiting adaptive bins for monocular depth estimation. *arXiv preprint arXiv:2204.00987*, 2022.
- [31] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Eur. Conf. Comput. Vis.*, pages 740–755, 2014.
- [32] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *Int. Conf. Learn. Representations*, 2020.
- [33] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *Int. Conf. Comput. Vis.*, 2021.
- [34] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Adv. Neural Inform. Process. Syst.*, 32, 2019.
- [35] Jiasen Lu, Christopher Clark, Rowan Zellers, Roozbeh Mottaghi, and Aniruddha Kembhavi. Unified-IO: A unified model for vision, language, and multi-modal tasks. *arXiv preprint arXiv:2206.08916*, pages 1–19, 2022.
- [36] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 2020.
- [37] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Eur. Conf. Comput. Vis.*, pages 746–760. Springer, 2012.
- [38] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. VL-Bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019.
- [39] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5693–5703, 2019.
- [40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Adv. Neural Inform. Process. Syst.*, 30, 2017.
- [41] Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. OFA: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. *Int. Conf. Mach. Learn.*, 2022.
- [42] Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. Solo: Segmenting objects by locations. In *Eur. Conf. Comput. Vis.*, pages 649–665, 2020.
- [43] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. SOIOv2: Dynamic and fast instance segmentation. *Adv. Neural Inform. Process. Syst.*, 33:17721–17732, 2020.
- [44] Zhendong Wang, Xiaodong Cun, Jianmin Bao, Wengang Zhou, Jianzhuang Liu, and Houqiang Li. Uformer: A general u-shaped transformer for image restoration. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 17683–17693, 2022.
- [45] Chen Wei, Wenjing Wang, Wenhan Yang, and Jiaying Liu. Deep retinex decomposition for low-light enhancement. *arXiv preprint arXiv:1808.04560*, 2018.
- [46] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *Eur. Conf. Comput. Vis.*, pages 472–487, 2018.
- [47] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Eur. Conf. Comput. Vis.*, pages 418–434, 2018.
- [48] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhiliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. *arXiv preprint arXiv:2111.09886*, 2021.
- [49] Wenhan Yang, Robby Tan, Jiashi Feng, Jiaying Liu, Zongming Guo, and Shuicheng Yan. Deep joint rain detection and removal from a single image. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.
- [50] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Multi-stage progressive image restoration. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 14821–14831, 2021.
- [51] Syed Waqas Zamir, Aditya Arora, Salman Hameed Khan, Hayat Munawar, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Learning enriched features for fast image restoration and enhancement. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 2022.
- [52] He Zhang and Vishal M. Patel. Density-aware single image de-raining using a multi-stream dense network. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.
- [53] He Zhang, Vishwanath Sindagi, and Vishal M. Patel. Image de-raining using a conditional generative adversarial network. *IEEE Trans. Circuits & Systems for Video Tech.*, 30(11):3943–3956, 2019.
- [54] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *Int. J. Computer Vision*, 2018.
- [55] Xizhou Zhu, Jinguo Zhu, Hao Li, Xiaoshi Wu, Hongsheng Li, Xiaohua Wang, and Jifeng Dai. Uni-Perceiver: Pre-training unified architecture for generic perception for zero-shot and few-shot tasks. *arXiv preprint arXiv:2112.01522*, pages 16783–16794, 2022.

## Appendix

### A. Additional Implementation Details

In this section, we provide additional details of the data preparation and pose-processing for different tasks. PyTorch-style pseudo-code is provided to better illustrate the implementation details. They can be done with a few operations in implementation. For each task, Painter doesn't involve more complex post-processing compared to the specialist methods.

**Semantic segmentation** As described in Section 3.1 of the main paper, we formulate different semantic categories using different colors in RGB space. To this end, we define the background and ignore areas as black, *i.e.*, pixels in color (0, 0, 0), and generate the colors for foreground categories using the pseudo-code elaborated in Figure S1.

During inference, to decode the output image to a single-channel ID map where each pixel represents a class ID, we compute the L1 distance between each output pixel and the pre-defined colors for each semantic category, and take the ID of the closest color as the predicted category. The pseudo-code for the post-processing is illustrated in Figure S2.

---

```
def generate_colors_dict(b, K):
    # b: the number of used values in a single channel
    # K: the number of classes

    m = 255 // b # get margin
    colors = []
    for class_id in range(K):
        # compute margin multiplier
        r_mult = class_id // b**2
        g_mult = (class_id % b) // b
        b_mult = class_id % b
        # compute r, g, b values
        r = 255 - r_mult * m
        g = 255 - g_mult * m
        b = 255 - b_mult * m
        colors.append((r, g, b))

    return colors
```

---

**Figure S1.** Pseudo-code for generating colors.  $//$ : floor division;  $\%$ : mod operation.

---

```
def forward(image, colors):
    # image: (H, W, 3)
    # colors: (K, 3), where K is the number of classes

    # get distance between pixels and pre-defined colors
    dist = (image.view(H, W, 1, 3) - colors.view(1, 1, K, 3))
        .abs().sum(-1) # (H, W, K)
    segm = dist.argmin(dim=-1) # (H, W)

    return segm
```

---

**Figure S2.** Pseudo-code of semantic segmentation post-processing.

**Keypoint detection** For keypoint detection, the output image consists of the R channel which denotes the class-agnostic heatmaps and the G/B channels that represent the keypoint categories. As illustrated in Figure S3, we convert the output image to a 17-channel heatmap, and follow the commonly used post-processing [39, 46] to obtain the final keypoint locations.

---

```
def forward(image, colors):
    # image: (H, W, 3)
    # colors: (K, 3), where K is the number of keypoints

    # r for heatmaps and gb for keypoint classes
    r = images[..., 0] # (H, W)
    gb = images[..., 1:] # (H, W, 2)

    # get keypoint class of each pixel
    dist = (gb.view(H, W, 1, 2) - colors.view(1, 1, K, 2))
        .abs().sum(-1) # (H, W, K)
    segm = dist.argmin(dim=-1) # (H, W)

    for idx in range(K):
        mask = segm == idx
        heatmap = mask * r # (H, W)
        heatmaps.append(heatmap)
    heatmaps = stack(heatmaps) # (K, H, W)

    return heatmaps
```

---

**Figure S3.** Pseudo-code of keypoint detection post-processing. `stack`: concatenates a sequence of tensors along a new dimension.

**Panoptic segmentation** As described in Section 3.2, we decompose the panoptic segmentation task into semantic segmentation and class-agnostic instance segmentation. During training, the semantic segmentation sub-task uses the same setting as the semantic segmentation on ADE-20K [54], except that we set the base  $b = 7$  when assigning colors. Similar to the color generation process of semantic segmentation, we generate colors for each location category [42] used in class-agnostic instance segmentation. The color of each instance mask is determined by the location of its center.

During inference, the semantic ID map can be obtained using the post-processing described in Figure S2, while the class-agnostic instance masks are generated by thresholding the distance between predicted colors and the pre-defined colors for location categories. Matrix NMS [43] is adopted to remove duplicate instance predictions. We apply the majority vote of pixels from the semantic prediction to get the semantic class for each instance mask, as illustrated in Figure S4. Finally, we follow Panoptic FPN [26] to merge the semantic segmentation and the instance segmentation predictions to obtain the panoptic segmentation results.

**Image restoration** The detailed statistics of the datasets that are used for image restoration are shown in Table S1.

Tasks	Deraining						Enhance	Denoising
Datasets	Rain14000 [15]	Rain1800 [49]	Rain800 [53]	Rain100H [49]	Rain100L [49]	Rain1200 [52]	Rain12 [29]	LoL [45]
Train Samples	11200	1800	700	0	0	0	12	SIDD [1]
Test Samples	2800	0	100	100	100	1200	0	485
Testset Rename	Test2800	-	Test100	Rain100H	Rain100L	Test1200	-	-
							15	40

**Table S1.** Dataset description for image restoration tasks.

(a) Patch merging			(b) Encoder			(c) Head type			(d) Loss function		
merging?	mIoU	mAcc	backbone	mIoU	mAcc	head	mIoU	mAcc	loss	mIoU	mAcc
✗	39.7	51.1	ViT-B	31.4	41.4	linear	38.6	50.2	$\ell_1$	40.6	52.5
✓	41.2	53.0	ViT-L	41.2	53.0	light	41.2	53.0	$\ell_2$	26.3	37.7

**Table S2.** Ablation study on ADE-20K semantic segmentation. (a) merging patch after three transformer blocks; (b) encoder; (c) head type; (d) loss function.

```
def forward(segm_dist, inst_masks):
    # segm_dist: (H, W, K), where K is the number of thing
    # classes
    # inst_masks: (N, H, W), where N is the number of
    # instances

    # turn distances to scores
    segm_scores = 1. - semseg_dist / max(semseg_dist)
    # majority vote
    class_probs = einsum("nhw,hwk->nk", inst_masks,
                         segm_scores) # (N, K)
    pred_classes = class_probs.argmax(dim=-1) # (N,)

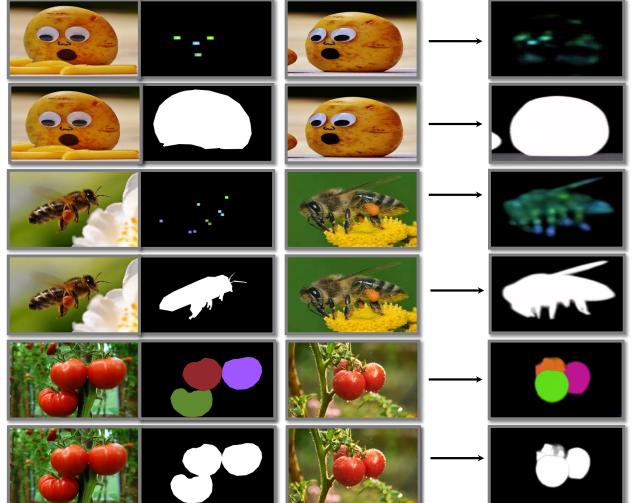
    return pred_classes
```

**Figure S4.** Pseudo-code of majority voting for labeling each instance mask. `einsum`: Einstein summation.

## B. Additional Results

We report the results of ablation experiments on several components of our framework, with a shorter schedule of 3k iterations and other hyper-parameters unchanged on semantic segmentation of ADE-20K.

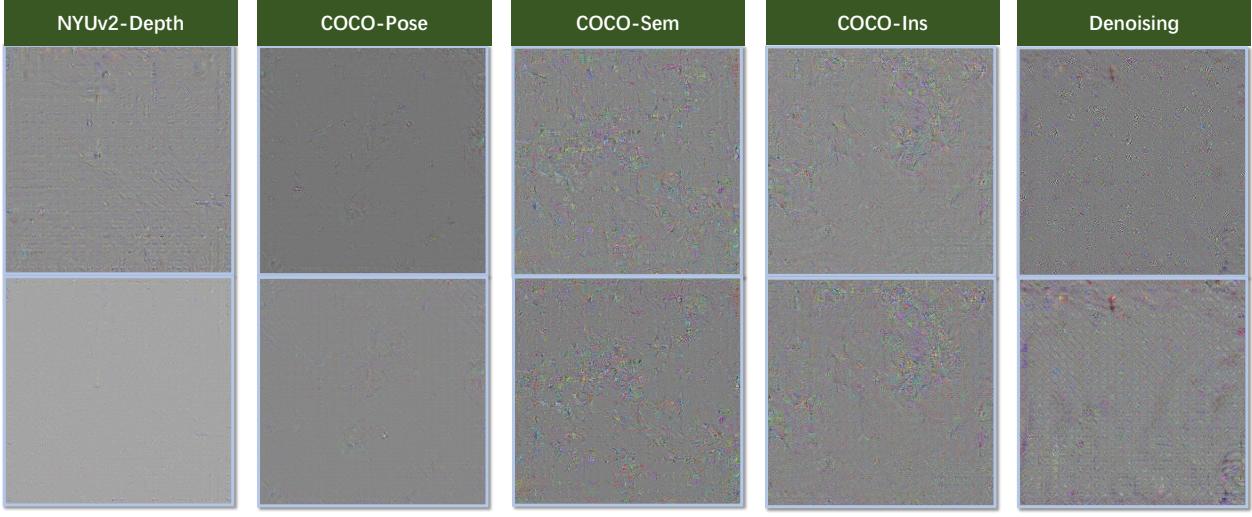
**Merging patches** During training, each input sample consists of both the input image and output image, which results in high memory cost and significantly slows down the training process. We reduce nearly half of the computation costs by merging the early features of the input image and the output image, *i.e.*, adding their features patch by patch after a three blocks. Table S2a shows that this new design even incurs performance increase. We argue that this design further provides the pixel-to-pixel correspondence between the input and its output via stacking them together. But in the original setting, these relationships need to be learned by the model, which will make the optimization more difficult especially in a short schedule.



**Figure S5.** More visualizations. The visualized tasks include keypoint detection, object segmentation and instance segmentation on potato, bee, and tomato.

**Encoder** We adopt standard Vision Transformer (ViT) [14] with different model sizes as the encoder, including ViT-base and ViT-large. Results are shown in Table S2b. We find that the model with ViT-L outperforms that with ViT-B by very large margins. This observation is intuitive, that generally larger models yield better performance. For generalist models, they can use more data but with less task-specific prior on method design, thus may require more model capacity than task-specific models.

**Head** We use a light three-layer head that consists of a linear ( $1 \times 1$  convolution) layer, a  $3 \times 3$  convolution layer, and another linear layer, to map the feature of each patch to its original resolution, *e.g.*,  $16 \times 16 \times 3$ . The feature of each



**Figure S6.** Visualizations of the learned prompts for different tasks. Each column denotes a prompt pair for a task. The first and second rows are input and output images respectively. For visualization, each prompt is normalized to an RGB image with values between 0 and 255. Different tasks show different patterns. Users can take the prompt images and feed them to the Painter model to enable the corresponding application.

patch is the concatenation of the 4 feature maps evenly sampled from the transformer blocks. As shown in Table S2c, the light head achieves clear gains over the baseline with only a linear layer.

**Loss function** Painter uses a simple pixel regression loss to learn all the tasks. In Table S2d, we compare different regression loss functions, including  $\ell_1$ ,  $\ell_2$ , and smooth- $\ell_1$ . We adopt smooth- $\ell_1$  by default as it achieves the best performance and is also more stable during training.

## C. Additional Visualization

In this section, we provide more visualizations. As shown in Figure S5, Painter performs in-context inference according to different prompt images. Note that Painter is never trained to solve these tasks during training, *e.g.*, keypoint detection of potato, object segmentation of bee, and instance segmentation of tomato.