Argus: A Compact and Versatile Foundation Model for Vision

Supplementary Material

A. Implementation Details

In this section, we provide detailed information about the adapter, the implementation of MTL library, the MTL algorithms compared in the main manuscript, and additional experiment setup.

A.1. Details on Adapter Components

As illustrated in Fig. 4, the adapter comprises several modules: a spatial prior module for spatial feature extraction, injector and extractor modules for interacting features with the ViT blocks, and a projection module for final feature aggregation and normalization. Below, we provide a detailed description of each module.

Spatial Prior Module. The spatial prior module (SPM) begins with a stem inspired by ResNet [26], consisting of three convolutional layers and a max-pooling layer. The output of the stem is followed by three 3×3 and 1×1 convolutional layers. These layers generate spatial features, $\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3\}$, at resolutions of $\frac{1}{8}$, $\frac{1}{16}$, and $\frac{1}{32}$, respectively. These spatial features are then flattened and concatenated to form spatial feature tokens $\mathcal{F}_{sp}$. These tokens are subsequently used as input to the $\mathcal{N}$ interaction blocks of the injector and extractor modules. To mitigate the negative transfer issue in multitask learning, we replace all the batch normalizations layers in this module with group normalizations layers.

Injector and Extractor. Each interaction block consists of an injector and an extractor, interacting with ViT blocks. For the i-th interaction block, we scale the ViT tokens $\mathcal{F}^i_{vit}$ and use them as the query for a cross-attention layer, while the spatial feature tokens $\mathcal{F}^i_{sp}$-either from the SPM or the prior extractor output-serve as the key and value. The injector performs an element-wise addition with the ViT features and then descales the cross-attention outputs, producing updated ViT features that are passed as input to the ViT blocks. After the ViT blocks, the extractor takes the scaled ViT features from these ViT blocks as key and value for another cross-attention layer, where the spatial feature tokens from the SPM (or the prior extractor output) acts as the query. The cross-attention layer output is then processed by a feed-forward network (FFN) to generate updated spatial feature tokens for the next interaction block. This iterative process enables the interactions between spatial and ViT tokens.

Projection Module. The projection module processes the final output of the backbone to generate features for decoders. It begins by performing element-wise addition of the final ViT features $\mathcal{F}^{\mathcal{N}}_{vit}$ with four feature maps, followed by group normalization layers. These feature maps consist of $\mathcal{F}_0$ (directly from the SPM) and $\{\mathcal{F}^{\mathcal{N}}_1, \mathcal{F}^{\mathcal{N}}_2, \mathcal{F}^{\mathcal{N}}_3\}$ (unflattened from spatial feature tokens $\mathcal{F}^{\mathcal{N}}_{sp}$ from the last interaction block). Group normalization ensures that each feature map is suitably normalized for the task-specific decoders, enhancing the adapter's capability across diverse vision tasks. These multiscale features $\{\mathcal{F}_0, \mathcal{F}^{\mathcal{N}}_1, \mathcal{F}^{\mathcal{N}}_2, \mathcal{F}^{\mathcal{N}}_3\}$ are then passed to the decoders.

A.2. Implementation Details of MTL Library

We developed a new modular and extensible multitask learning (MTL) package to facilitate the experiments in this paper. The library is designed to support training across various computer vision tasks, covered by our vision foundation model (VFM) `Argus`. For each specific task, the package integrates a corresponding task-specific library responsible for dataset loading, model construction, and task evaluation. Specifically, we utilized the following integrations: `mmdetection`[2] for detection, instance segmentation, and panoptic segmentation; `mmsegmentation`[3] for depth estimation, surface normal, boundary deteciton, semantic segmentation, human parsing, and saliency detection; `mmpose`[4] for pose estimation; and `mmclassification`[5] for classification.

This modular architecture allows for seamless integration of additional task-specific libraries, enabling the package to support new tasks with minimal effort. Users can either incorporate new task-specific modules or leverage the existing integrations.

To improve the training efficiency, we implemented the automatic mixed precision (AMP). This optimization reduces memory consumption and accelerates training. These efforts enable efficient training of the five core tasks in MTL using simple distributed data parallel (DDP) without requiring more complex techniques such as model parallelism or gradient checkpointing.

The package also supports multiple MTL algorithms, including FAMO [42] and GradNorm [11].

A.3. Implementation Details of MTL Algorithms

We introduce our MTL formulation in Section 3. For the weighting factor λ_t in Equation 1, we set it heuristically based on training loss scales in multitask learn-

[2] https://github.com/open-mmlab/mmdetection
[3] https://github.com/open-mmlab/mmsegmentation
[4] https://github.com/open-mmlab/mmpose
[5] https://github.com/open-mmlab/mmclassification

ing. For example, if the loss for pose/seg/cls/det is around 0.001/0.25/3/50 in the middle of training (set $\lambda_t = 1$ as the default), then we set $\lambda_{\text{pose}} = 1000$, $\lambda_{\text{seg}} = 3$, $\lambda_{\text{cls}} = 1$, and $\lambda_{\text{det}} = 0.1$ to scale the task losses roughly to the same range of $[0.5, 5]$.

We also provide the detailed configurations of the other MTL algorithms compared in the ablation study in Sec. 4.3 below.

FAMO. The Fast Adaptive Multitask Optimization (FAMO) [42] is a loss balancing MTL algorithm that is desgined based on the principle that at each step, all task losses should decrease at the same rate. We implemented FAMO in our MTL framework based on their official codebase[6]. We follow the default configurations in their codebase. The learning rate for the Adam optimizer of weighting parameter is 0.025, and the weight decay is 0.001. The task weights are normalized to have the sum equal to the number of tasks.

GradNorm. The GradNorm algorithm [11] balances the gradients of different tasks based on the principle that the gradient from different tasks should share a common scale. Since computing the exact gradient w.r.t. the shared parameters is computationally expensive, we approximate the gradient norm by the norm of the gradient of the feature maps. In our multi-input case, different tasks have different input images with different batch sizes and different resolutions and therefore have different feature maps. In order to counter these differences, we first compute the norm of the gradient of the feature maps, and then average them over batch size and spatial resolution.

A.4. Details on Experiment Setup

By default, we use the AdamW [48] optimizer with a learning rate of 0.0002, a weight decay 0.05, and a Cosine Annealing learning rate scheduler [49]. We use batch size of 16 for object detection, instance segmentation, panoptic segmentation, semantic segmentation on ADE20K dataset, depth estimation, surface normal, and object boundary detection. For saliency detection, human parsing, and semantic segmentation on the NYUv2 dataset, a batch size of 32 is used. Additionally, we use a batch size of 128 for pose estimation on the COCO dataset and 256 for classification on the ImageNet dataset.

For the another 5 tasks selected for ablation in Sec. 4.3, they are instance segmentation on ADE20K, depth estimation and boundary detection on NYUv2, and surface normal and human parsing on PASCAL-context.

[6]https://github.com/Cranial-XIX/FAMO

B. Details on Supported Tasks

Object Detection. Object detection involves identifying and localizing objects within an image by predicting their bounding boxes and class labels. Instance segmentation is often trained together using the same detector such as Mask R-CNN [27] or Mask DINO [35] to extends this task by providing pixel-level segmentation masks for each detected object. We train object detection and instance segmentation together using Mask DINO [35] on COCO dataset. Mask DINO is an advanced architecture designed for end-to-end object detection and segmentation. It jointly optimizes the detection head and segmentation head through shared representations, enabling the simultaneous learning of bounding boxes, class labels, and segmentation masks. We follow the standard configurations and loss functions provided by Mask DINO. The object detection head predicts bounding boxes and class scores, while the instance segmentation head generates pixel-level masks.

Pose Estimation. Pose estimation aims to detect and predict the positions of keypoints of objects (*e.g.* human body, face, animal) within an image. We adopt the top-down keypoint heatmap approach, which has been shown to achieve superior performance in human body pose estimation [77]. Given the bounding box of each individual human body (which can be reliably obtained by `Argus`'s object detection decoder), the top-down model transforms the keypoint detection into a heatmap estimation problem. We use Dense Prediction Transformer (DPT) [59] head with regression loss of keypoints to train this task.

Classification. Image classification involves assigning a label to an entire image based on its content. We apply an attentional pooler [83] to the multiscale spatial tokens from our image encoder to produce a single holistic representation of images. We then feed this token to a linear projection layer followed by a Softmax activation to form the classification decoder, which is trained using the conventional cross-entropy loss by the manual class labels.

Semantic Segmentation, Saliency Detection, Human Parsing, and Object Boundary Detection. These four tasks involve detailed image analysis and segmentation. Semantic segmentation classifies each pixel in an image into a predefined category, providing a granular understanding of the scene. Human parsing focuses on identifying and labeling various body parts within an image, allowing for precise body part recognition. Saliency detection aims to highlight the most visually prominent areas of an image, directing attention to key regions. Object boundary detection involves identifying and outlining the boundaries of objects, delineating their shapes and contours. For these tasks,

Table 10. Evaluation results on Single Task, multi-task adaptation, and comparisons with existing models.

Methods	COCO			ImageNet	ADE20K
	AP_b	AP_m	AP_k	Top-1	mIoU
Single Task	56.7	50.4	76.4	86.2	55.4
Multi-task (Ours)	58.6	51.8	77.0	86.3	56.5
Adaptation (Ours)	58.8	52.0	77.1	86.3	56.7

we employ UPerNet [76], a powerful decoder known for generating high-resolution segmentation masks. Except for saliency detection, the UPerNet decoder is supplemented by an auxiliary Fully Convolution Network [47], which is only used during training. We use pixel-wise cross-entropy loss as the training objective for all these tasks.

Instance and Panoptic Segmentation. These two tasks involve identifying and segmenting individual objects and their boundaries within an image. We use Mask2Former [13] as the decoder for both tasks. For Mask2Former, we adopt the original pipeline, where low-resolution features from the VFM backbone are upsampled and decoded by a pixel and transformer decoder, respectively. Besides, we use a combination of binary cross-entropy (BCE) loss and dice loss as the mask losses, together with classification loss for training.

Depth and Surface Normal Estimation. The goal of these tasks is to predict the depth and orientation of surfaces in an image. For both tasks, we use a Dense Prediction Transformer (DPT) [59] as the decoder. For depth estimation, we use the pixel-wise scale-invariant depth loss from AdaBins [3] together with the multiscale gradient-matching loss from MegaDepth [37]. The DPT decoder with a simple L2 loss is used to train our surface normal prediction model, as done by other relevant works [79, 87]. The output surface normal vectors are consistently normalized w.r.t. the L2-norm.

Anomaly Detection. We adopt SimpleNet [46] as our anomaly detection decoder. In SimpleNet, a feature adapter that preprocesses features from selected layers of the backbone to obtain normal representations. During training, an anomalous feature generator simulates anomaly features by sampling random Gaussian noise and adding it to the normal features. These normal and simulated anomalous features are then processed by a discriminator, which serves as a normality scorer to evaluate the normality of each pixel location. During testing, the anomalous feature generator is discarded, and the feature adapter and discriminator are employed to perform anomaly detection inference.

C. Dataset Statistics

Tab. 11 summarizes the datasets used in `Argus`.

D. More Ablation Studies

D.1. Fairness of Comparison

In Tab. 1 and Tab. 2, we compare Argus with existing works. Existing foundation models lack standardization in training datasets and tasks during pre-training. Argus uses less than 1.8M images for training (stages 1 & 2), similar to MAE and GLID, while other models require significantly larger datasets (e.g., 4M uses 12M images, Florence uses 100M image-text pairs, and Florence-2 uses 126M images). Despite our smaller model size and training data, Argus achieves compelling performance across 12 vision tasks. Most comparisons are based on seen datasets, except Florence-2-B, for which we report performance on both unseen and seen datasets, denoted as 'Florence-2-B' and 'Florence-2-B (fine-tuned)', respectively.

Additionally, we mark the models using unified decoders in Tab. 1 and Tab. 2. Different methods inherently use different decoders, and prior works like BEiT-3 [72] and InternImage [73] also compare models with varying decoders. All the decoders of DINOv2 are the same as Argus. Furthermore, we also provide performance of Argus with different object detection decoders in Tab. 6, including Mask R-CNN, Mask DINO, and CO-DETER for references. Argus outperforms MAE with the same Mask R-CNN decoder.

D.2. Single Task and Core Task Adaptation

Tab. 10 compares Argus with multi-task learning (MTL) and training each task separately (Single Task) with the adapter and decoder. Argus with MTL outperforms single task on all these datasets. Besides, task-specific adaptation is only applied to the other 7 tasks, as multitask pre-training already produces strong performance on the 5 core tasks by directly training their decoders. As shown in Tab. 10, further adaptation on these 5 tasks only results in marginal improvements.

D.3. Compare with More Models

We compare `Argus` with multitask learning methods and several models specialized on the 7 tasks that are less covered in foundation models, extending the comparison presented Tab. 2. In Tab. 12, we compare `Argus` with multitask models such as TaskExpert [81] and InvPT [79], as well as specialized models, including ViTPose [77] for pose estimation and DRAEM [86] for anomaly detection. Remarkably, `Argus` not only supports all these tasks but also consistently outperforms these models.

Dataset	Task	Multitask Pretraining	Task-Specific Adaptation	Evaluation	Size (Train / Test)	Metric
COCO	Object Detection	Yes	No	Yes	117 266 / 4952	mAP
	Instance Segmentation	Yes	No	Yes	117 266 / 4952	mAP
	Pose Estimation	Yes	No	Yes	149 813 / 6352	AP, AR
ImageNet-1K	Classification	Yes	No	Yes	1 281 167 / 50 000	Top-1 Acc
ADE20K	Semantic Segmentation	Yes	No	Yes	20 210 / 2000	mIoU
	Instance Segmentation	No	Yes	Yes	13 385 / 1353	mAP
	Panoptic Segmentation	No	Yes	Yes	17 277 / 1743	PQ
NYUv2	Depth Estimation	No	Yes	Yes	24 231 / 654	RMSE
	Object Boundary Detection	No	Yes	Yes	795 / 654	odsF
	Surface Normal Estimation	No	Yes	Yes	795 / 654	mErr
PASCAL-Context	Human Parsing	No	Yes	Yes	1736 / 1853	mIoU
	Saliency Detection	No	Yes	Yes	6352 / 5105	maxF
MVTecAD Transistor	Anomaly Detection	No	Yes	Yes	213 / 100	I-AUROC
MVTecAD Metal Nut	Anomaly Detection	No	Yes	Yes	220 / 115	I-AUROC
MVTecAD Screw	Anomaly Detection	No	Yes	Yes	320 / 160	I-AUROC
MVTecAD Leather	Anomaly Detection	No	Yes	Yes	245 / 124	I-AUROC
Person Camera Security	Object Detection	No	No	Yes	- / 430	AP
Pascal VOC 2012 Person	Object Detection	No	No	Yes	- / 1789	AP
Detecting Cars	Object Detection	No	No	Yes	- / 116	AP

Table 11. Summary of the datasets used for `Argus`.

Methods	# Params	Pose Estimation		Depth Estimation	Boundary Detection	Surface Normal	Human Parsing	Saliency Detection	Anomaly Detection
		COCO		NYUv2	NYUv2	NYUv2	PASCAL-C	PASCAL-C	MVTecAD-4
		$AP_k \uparrow$	AR ↑	RMSE ↓	odsF ↑	mErr ↓	mIoU ↑	maxF ↑	I-AUROC ↑
DRAEM [86]	69M	-	-	-	-	-	-	-	96.4
ViTPose (ViT-B) [77]	86M	75.8	81.1	-	-	-	-	-	-
TaskExpert [81] (ViT-B)	347M	-	-	-	-	-	67.4	85.0	-
InvPT [79]	176M	-	-	0.598	76.1	20.5	62.7	84.2	-
Argus	100M	**77.0**	**81.8**	**0.290**	**76.7**	**18.6**	**77.8**	**97.2**	**99.3**

Table 12. Performance comparison of Argus with multitask models and specialized models over 7 vision tasks less covered in foundation models on COCO [40], NYUv2 [33], PASCAL-C [55] and MVTecAD-4 [2] datasets.

D.4. Generalization on the Unseen Datasets

Object Detection. To demonstrate the generalizability of Argus, we compare the performance of Argus with 4M [53] and Florence-2 [75] on unseen data in Tab. 8. Here we provide sources of the datasets: Pascal VOC 2012 Person [7], Person Camera Security [8] and Detecting Cars [9] datasets from Roboflow Universe.

Image Classification. We adapt Argus to 11 fine-grained object recognition benchmark datasets [33] to investigate the generalizability of our learned feature representations to unseen data. In Tab. 14, we take DINOv2 with the default linear head as a baseline [57], and provide a new result using the same classification decoder as ours, denoted as DINOv2*, for a fair comparison. The performance superiority of DINOv2* over DINOv2 indicates the effectiveness of the classification decoder of our Argus. Besides, Argus yields better performances than DINOv2* across most datasets, demonstrating its generalization capability to unseen data.

D.5. Validation Performance Curves

In this section, we provide the validation performance curves of various experiment settings.

Freeze ViT *vs*. Training All Paramters. In Fig. 5, we show the validation performances of freezing ViT *vs*. without freezing ViT (*i.e.* training all parameters) in multitask pretraining of 5 tasks. We observe that freezing ViT converges much faster and achieve higher performance in the end of training. Training all parameters have significantly lower performance in the object detection, instance segmentation, semantic segmentation, and classification tasks. Only the pose estimation performance is comparable to the freeze setting.

BN *vs*. GN. In Fig. 6, we show the validation performances of BN *vs*. GN in multitask pretraining of 5 tasks. We observe that GN consistently outperforms BN in all tasks. Compared with the default configuration with BN (SyncBN in the distributed training case), using GN can mitigate the negative transfer in the MTL stage.

Overfitting in 11 MTL tasks We plot both the training loss and the validation performance of multitask pretraining of 11 tasks in Figs. 7 and 8. We observe that for tasks with small quantity of training data, the model tends to overfit, which is reflected by the performance drop in the validation set and the continuous decrease of the training loss.

E. Qualitative Study

E.1. Image Classification

Since the top 2 competitors, Florence [85] and Uni-perceiver v2 [36] in Tab. 1, are not open-source on the ImageNet classification benchmark dataset. For qualitative evaluation, we compare Argus with the next best competitor, DINOv2[10] [57]. Specifically, we visualize different model's spatial attention corresponding to their predicted classes for object recognition by GradCAM [62]. As shown in Fig. 9, Argus performs well at identifying salient objects in images and categorizing them accurately in some challenging cases where DINOv2 might lose its focus potentially due to noisy backgrounds, poor illumination conditions and *etc*.

E.2. Object detection

Similar to the image classification case, since the top 2 competitors, Florence and Uni-perceiver v2 in Tab. 1, are not open-source, we compare the qualitative results of Argus

[7]https://public.roboflow.com/object-detection/pascal-voc-2012/1
[8]https://universe.roboflow.com/chinh/person_camera_security1/dataset/21
[9]https://universe.roboflow.com/cars-uqebl/detecting-cars-9jzqm/dataset/5
[10]DINOv2's classification head is from their official GitHub repository https://github.com/facebookresearch/dinov2.

MTL Tasks	COCO			ImageNet	ADE20K			NYUv2			PASCAL-Context		
	AP_b	AP_k	AP_m	Top-1	mIoU	AP	PQ	RMSE ↓	odsF	mIoU	mErr ↓	mIoU	maxF
5 Tasks	57.7	51.1	75.2	85.1	55.6	37.9	45.4	0.314	75.81	64.83	19.20	77.79	96.97
8 Tasks	57.3	50.9	74.8	84.8	56.6	34.2	44.9	0.362	75.82	64.31	19.06	77.48	96.88
11 Tasks	57.6	50.9	74.8	84.9	56.3	33.0	43.0	0.363	75.02	61.48	19.13	77.88	96.05
Another 5 Tasks	54.9	48.9	73.0	82.9	54.8	33.6	45.2	0.348	74.90	63.91	18.81	77.74	96.97

Table 13. Multitask pretraining on more tasks and an alternative set of 5 tasks. Tasks marked in blue means these tasks are not considered in MTL pretraining (stage 1 in Fig. 3) and only covered in the task-specific adaptation (stage 2 in Fig. 3).

Dataset	Food101	CIFAR10	CIFAR100	SUN397	Stanford Cars	FGVC Aircraft	VOC2007	DTD	Oxford Pets	Caltech101	Flowers102	Average ↑
Metric	Acc ↑	Acc ↑	Acc ↑	Acc ↑	Acc ↑	Mean Per-Class Acc ↑	mAP ↑	Acc ↑	Mean Per-Class Acc ↑	Mean Per-Class Acc ↑	Mean Per-Class Acc ↑	
DINOv2 [57]	92.8	98.7	91.3	77.3	88.2	79.4	88.2	83.3	96.2	96.1	99.6	90.1
DINOv2*	93.9	97.3	86.3	78.8	94.0	84.0	95.0	84.5	95.6	96.4	99.7	91.4
Argus	93.7	97.6	86.7	80.4	93.9	83.6	95.8	84.7	96.4	97.2	99.7	**91.8**

Table 14. Comparison of classification performance on unseen datasets. We adapt to 11 fine-grained object recognition benchmark datasets [33]. We take the ViT-B/14 variant of DINOv2 with the linear head as the baseline. For a fair comparison, we build an additional baseline by replacing the linear head of DINOv2 with the classification decoder of our Argus, and denote it as DINOv2*.

with Florence-2 and 4M. We used their open-source pretrained models [11] to run object detection on COCO test images and visualize the results in Fig. 10. Argus demonstrates superior object detection performance compared to Florence-2 and 4M, particularly in challenging scenarios involving small objects or scenes with low-lighting conditions.

E.3. Pose Estimation

We compare the qualitative result of Argus with the ViT-Pose [77] as GLID [43] is not open-sourced. We used the open-source pretrained models [12] to run pose estimation on COCO test images and visualize the results in Fig. 11. Argus has more accurate elbow and hip joint estimation compared to ViT-Pose, which demonstrates Argus's strong performance in pose estimation.

E.4. Anomaly detection

We compare Argus with the DRAEM [86]. We used the open-source pretrained models [13] to run anomaly detection on MVTecAD test images and visualize the results in Fig. 12. Argus has less noise in the predicted anomalous region compared to DRAEM, which is critical in detecting anomalies. This demonstrates Argus's superior anomaly detection performance.

E.5. Depth Estimation

We compare Argus with DINOv2 [57] and InvPT [79]. We used the open-source pretrained models [14] to run depth estimation on NYUv2 [65] test images and visualize the results in Fig. 13. Argus outperforms both DINOv2 and InvPT in fine-grained details and prediction sharpness for depth prediction tasks, offering substantial advantages in producing more accurate and high-resolution depth maps. Unlike DINOv2, which sometimes struggles with capturing intricate textures in complex scenes, Argus excels in discerning minute variations in object surfaces and spatial transitions, leading to a significantly enhanced level of detail. Additionally, while InvPT has made strides in depth prediction, Argus's architecture is optimized to prioritize depth precision, minimizing blurriness and artifacts in the output. This makes Argus particularly suited, as it captures nuanced depth gradients that competing models tend to overlook. Consequently, Argus not only delivers more visually pleasing results but also enhances performance in tasks that rely heavily on depth accuracy and fine-grained spatial awareness.

[11]The models are downloaded from their official Hugging Face repositories: https://huggingface.co/microsoft/Florence-2-base, https://huggingface.co/EPFL-VILAB/4M-21_B.

[12]The models are downloaded from their official GitHub repositories: https://github.com/ViTAE-Transformer/ViTPose

[13]The models are downloaded from their official GitHub repositories: https://github.com/VitjanZ/DRAEM

[14]The models are downloaded from their official GitHub repositories: https://github.com/zqy1/InvPT/, https://github.com/facebookresearch/dinov2

E.6. Object Boundary Detection

We compare Argus with InvPT [79]. We used the open-source pretrained models [15] to run object boundary detection on NYUv2 [65] test images and visualize the results in Fig. 14. Argus significantly outperforms InvPT, producing more accurate predictions for the object boundary detection task. Its results demonstrate higher odsF (as shown in Tab. 2) in identifying object edges, achieving clearer and more reliable boundary delineation. Argus consistently surpasses InvPT in odsF metric, making it the preferred choice for tasks requiring detailed and precise object boundary detection.

E.7. Human Parsing

Fig. 18 illustrates a qualitative comparison between Argus and DINOv2 [57] on the human parsing task, using the PASCAL-Context dataset [55]. The figure illustrates that Argus produces more accurate and detailed segmentation masks, particularly in complex and challenging scenarios. For example, in cases where multiple objects or body parts are closely positioned or occluded, Argus successfully delineates finer details and preserves the boundaries, whereas DINOv2 often struggles with misclassification or merging adjacent regions.

E.8. Saliency Detection

Fig. 19 showcases the qualitative comparison between Argus and DINOv2 [57] on the salient object detection task using the PASCAL-Context dataset. The figure demonstrates that Argus consistently generates more precise and accurate saliency maps, effectively highlighting the most prominent objects in the scene. In contrast, DINOv2 frequently produces less defined or overly diffuse saliency maps, sometimes failing to distinguish the primary objects from the background or adjacent elements.

E.9. Instance Segmentation

We provide a qualitative comparison of Argus with the baseline DINOv2 [57] model in Fig. 15 for the instance segmentation task. We use images from the test set of the COCO dataset [40] for this comparison. As Argus outperforms the baseline model in quantitative performance (AP_m) in Tab. 1, we also observe a noticeable improvement in the quality of generated masks and the model's confidence in object classification. In comparison to DINOv2, we find that Argus achieves better success in recognizing small objects and generates segmentation maps that closely align with the ground truth.

[15]The models are downloaded from their official GitHub repositories: https://github.com/zqy1/InvPT/

E.10. Panoptic Segmentation

For panoptic segmentation task, we provide qualitative comparison of Argus with the baseline DINOv2 [57] model in Fig. 16. We use validation images from the ADE20K dataset in this experiment. Similar to the instance segmentation task, we find that Argus not only outperforms the baseline model in quantitative performance (PQ) but also achieves better qualitative results, e.g., often identifying objects that the baseline model completely failed to recognize in its panoptic segmentation maps.

E.11. Semantic Segmentation

We provide a qualitative comparison of Argus with the baseline DINOv2 [57] model in Fig. 17 for the semantic segmentation task. Similar to panoptic segmentation, we use validation images from the ADE20K dataset for this comparison. In conjunction with a significantly better quantitative performance (mIoU), we observe that Argus achieves significantly better qualitative results than the baseline, e.g., generating more accurate segmentation maps in crowded scenes.

E.12. Surface Normal Estimation

As shown in Fig. 20, Argus demonstrates a clear advantage over DINOv2 [57] in the surface normal prediction task using the NYUv2 dataset, particularly in handling complex scenes with intricate structures. The surface normal maps generated by Argus are not only more accurate but also capture fine details with remarkable precision, preserving the subtle variations in surface orientation across objects. These finer details are especially evident in environments with detailed architectural features or cluttered settings, where Argus excels at accurately mapping surface orientations. On the other hand, DINOv2's predictions tend to lose clarity in scenes with high geometric complexity, producing surface normal maps that are less precise and often fail to distinguish finer structural elements.

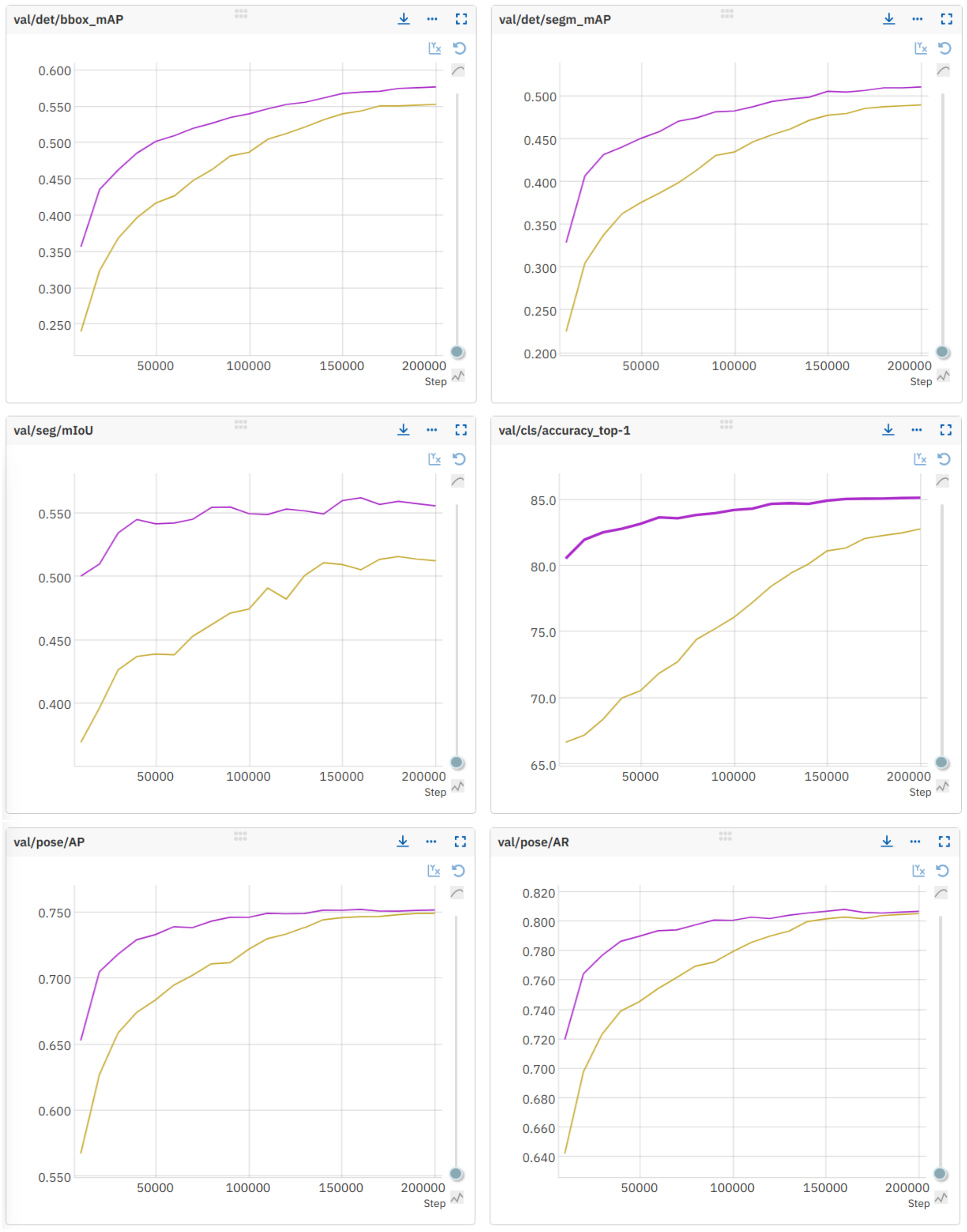


Figure 5. The validation performance of freezing ViT (the purple curve) *vs*. without freezing ViT (*i.e*. training all parameters, the yellow curve) in multitask pretraining of 5 tasks.Freezing ViT converges much faster and achieves higher performance in the end of training. The tasks are annotated with labels: object detection (det/bbox_mAP), instance segmentation (det/segm_mAP), semantic segmentation (sem/mIoU), classification (cls/accuracy_top-1), pose estimation (pose/AP, pose/AR).

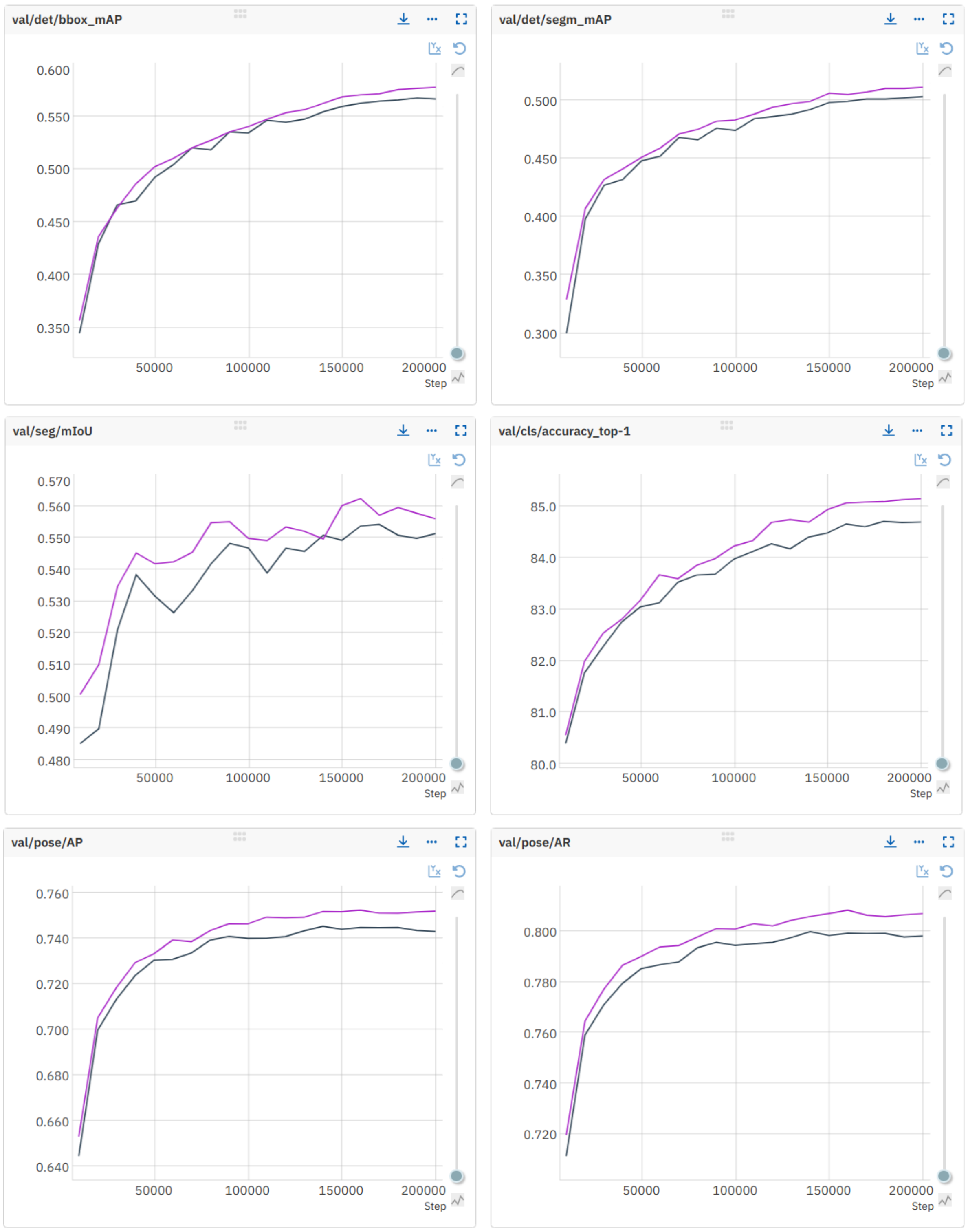


Figure 6. The validation performance of BN (the gray curve) *vs*. GN (the purple curve) in the 5-task MTL training. GN consistently outperforms BN in all tasks. The tasks are annotated with labels: object detection (det/bbox_mAP), instance segmentation (det/segm_mAP), semantic segmentation (sem/mIoU), classification (cls/accuracy_top-1), pose estimation (pose/AP, pose/AR).

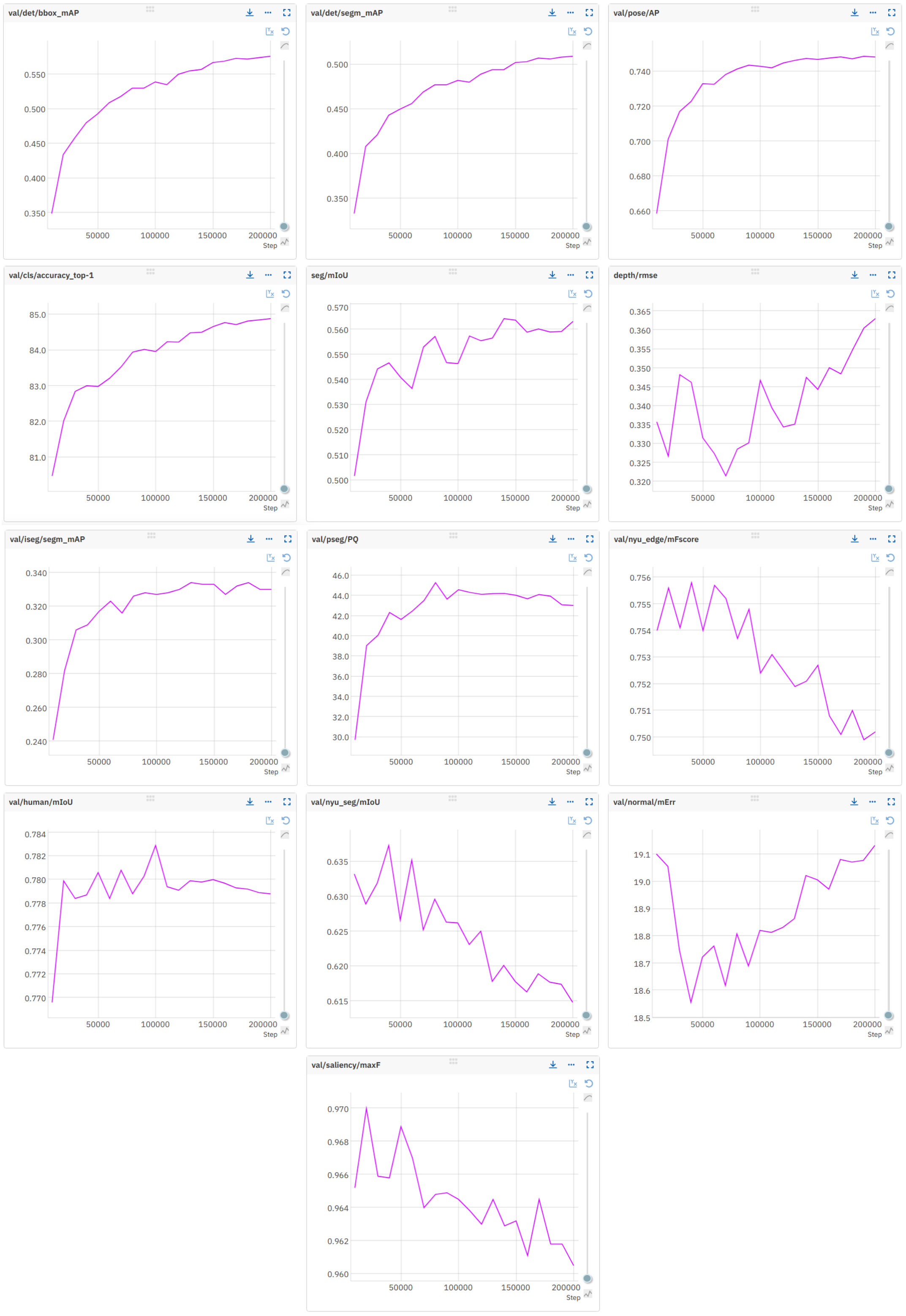


Figure 7. Validation performances of multitask pretraining of 11 tasks. We observe that the validation performance decreases in the later stage of the training for several tasks, including depth estimation (depth/rmse), object boundary detection (nyu_edge/mFscore), human parsing (human/mIoU), semantic segmentation on NYUv2 (nyu_seg/mIoU), surface normal (normal/mErr), and saliency detection (saliency/maxF). As illustrated in Fig. 8, where the training losses for all tasks continue to decrease, it suggests that the model may be overfitting on these tasks.

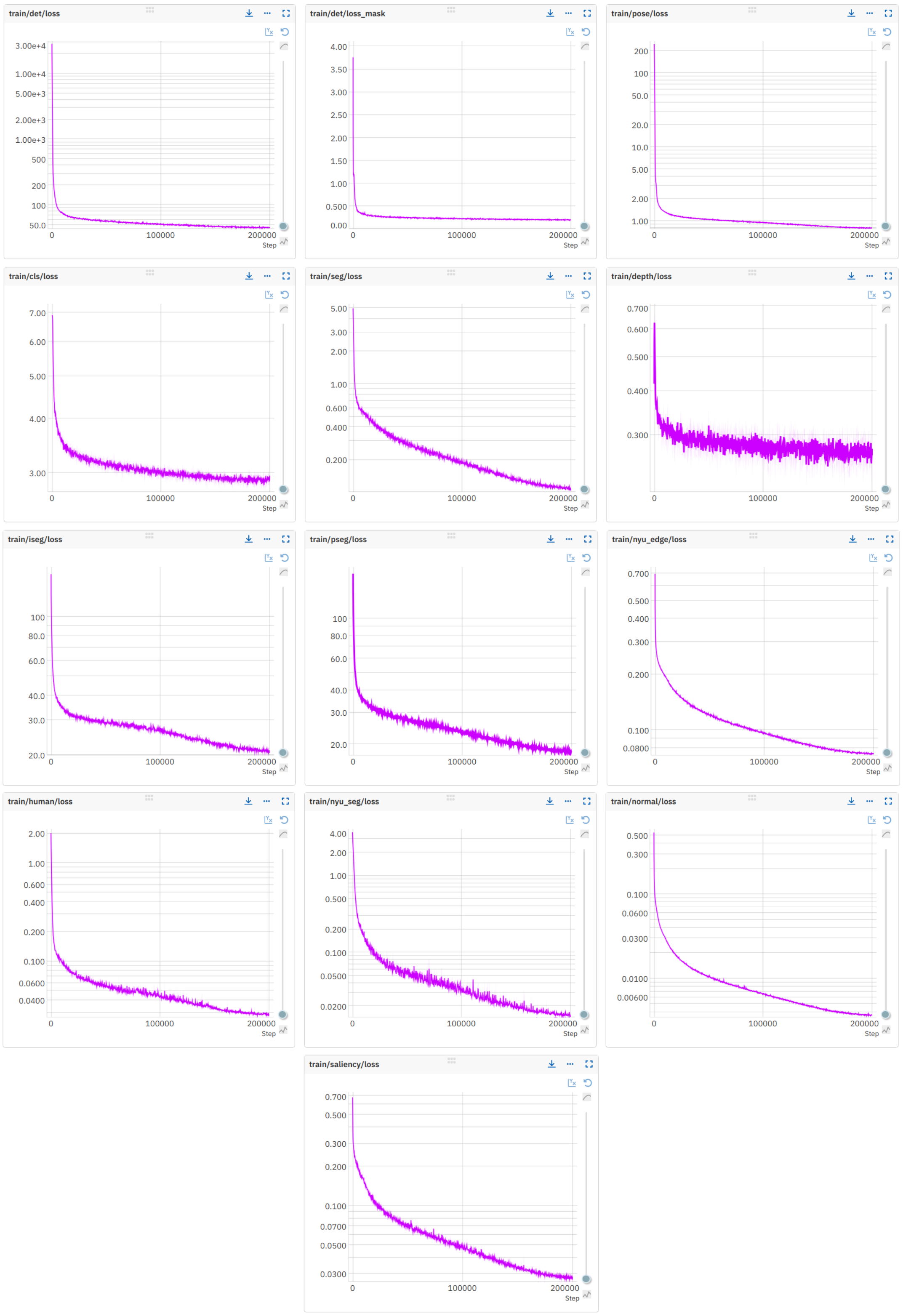


Figure 8. Training losses of multitask pretraining of 11 tasks. We observe that the training loss for all tasks continuously decrease throughout training. However, as shown in Fig. 7, the validation performance for several tasks decline in the later stages of training. These results indicate that the model may be overfitting on these tasks.

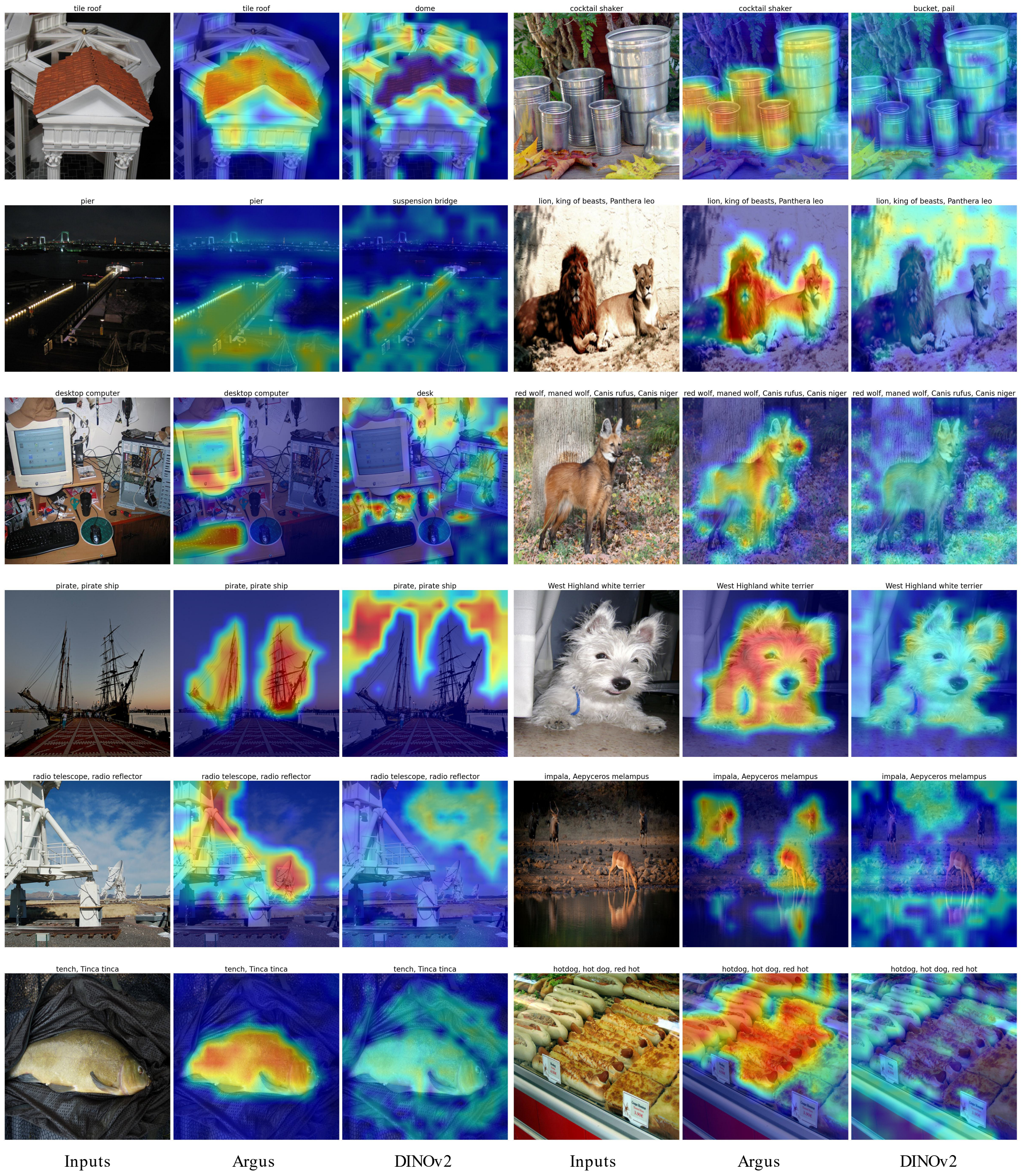
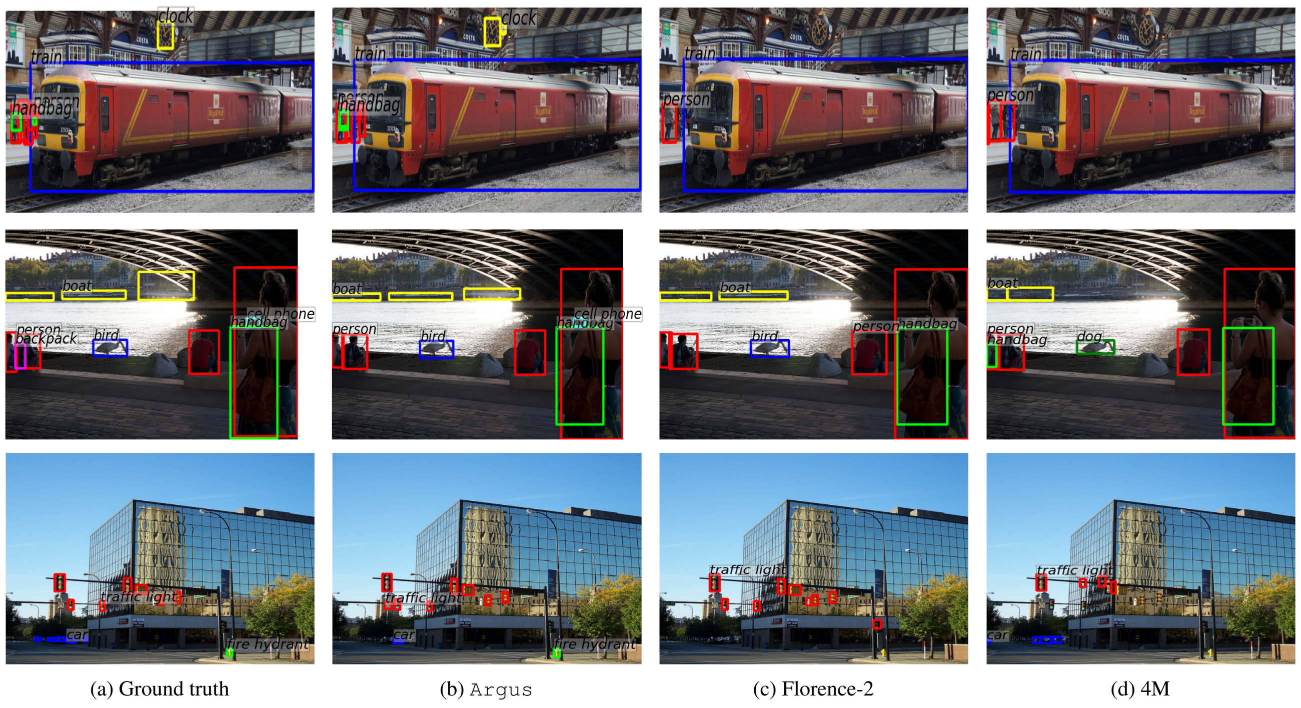
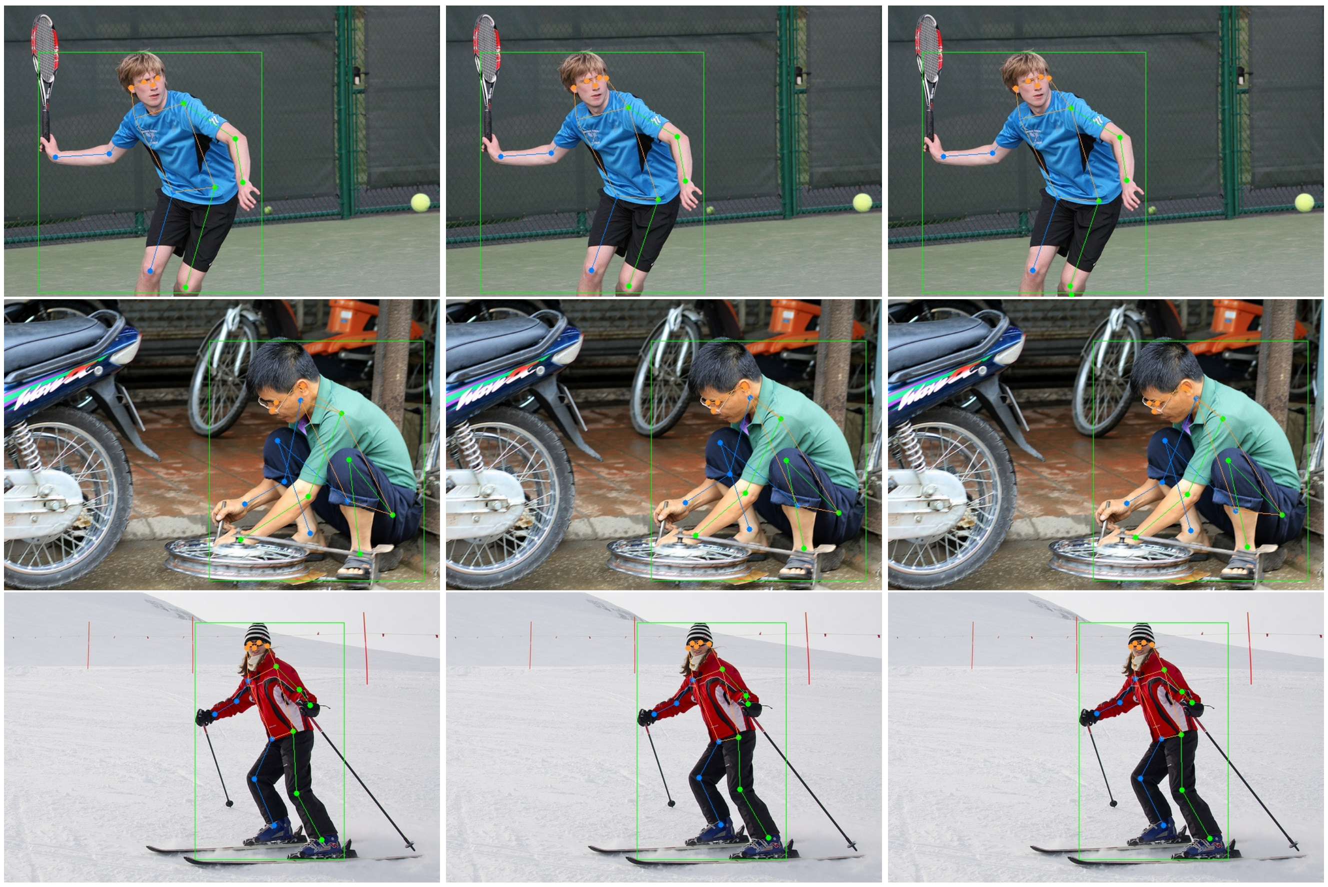

tile roof
tile roof
dome
cocktail shaker
cocktail shaker
bucket, pail
pier
pier
suspension bridge
lion, king of beasts, Panthera leo
lion, king of beasts, Panthera leo
lion, king of beasts, Panthera leo
desktop computer
desktop computer
desk
red wolf, maned wolf, Canis rufus, Canis niger
red wolf, maned wolf, Canis rufus, Canis niger
red wolf, maned wolf, Canis rufus, Canis niger
pirate, pirate ship
pirate, pirate ship
pirate, pirate ship
West Highland white terrier
West Highland white terrier
West Highland white terrier
radio telescope, radio reflector
radio telescope, radio reflector
radio telescope, radio reflector
impala, Aepyceros melampus
impala, Aepyceros melampus
impala, Aepyceros melampus
tench, Tinca tinca
tench, Tinca tinca
tench, Tinca tinca
hotdog, hot dog, red hot
hotdog, hot dog, red hot
hotdog, hot dog, red hot
Inputs
Argus
DINOv2
Inputs
Argus
DINOv2

Figure 9. Qualitative comparison between `Argus` and DINOv2 [57] on ImageNet dataset [61]. We visualize the image regions that different models focused on when making class predictions using GradCAM [62]. The manual class labels and the corresponding predictions from `Argus` and DINOv2 are provided on top of each image. `Argus` can perform well in some challenging cases where DINOv2 is distracted by noisy backgrounds, poor illumination conditions and *etc*.



(a) Ground truth (b) `Argus` (c) Florence-2 (d) 4M

Figure 10. Qualitative comparisons of `Argus` against Florence-2 [75] and 4M [53] on object detection. We visualize the results of these methods on COCO [40] test images. `Argus` demonstrates superior object detection performance compared to Florence-2 and 4M, particularly in challenging scenarios involving small objects or scenes with low-lighting conditions.



(a) Ground truth (b) `Argus` (c) ViT-Pose

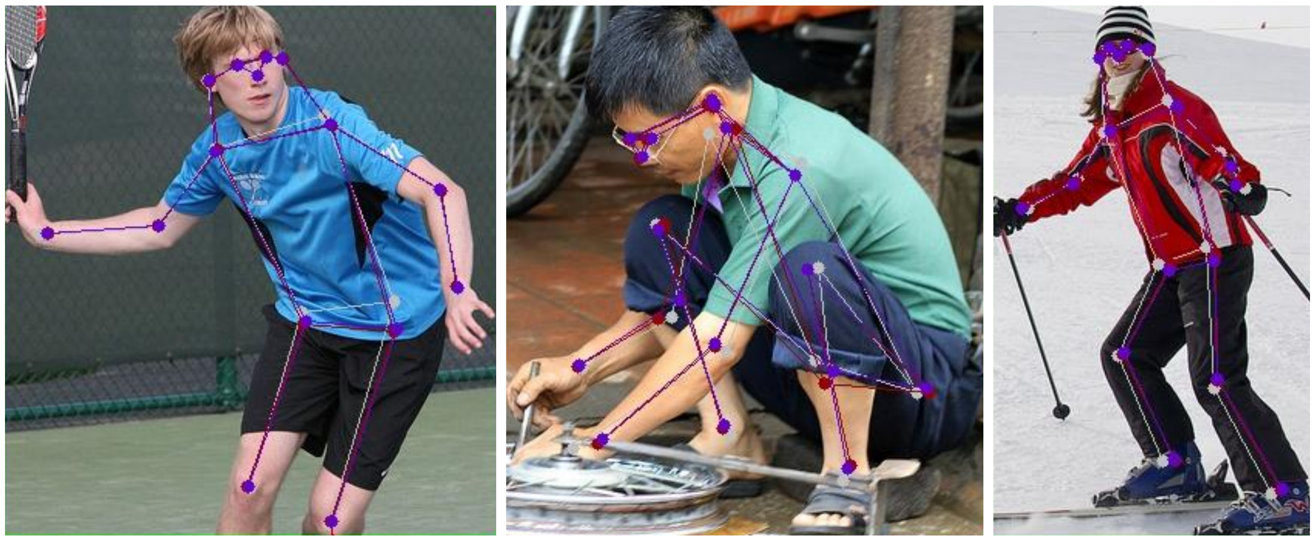


Figure 11. Qualitative comparisons between `Argus` and ViT-Pose [77]. We visualize the results of these methods on COCO test images. `Argus` demonstrates strong pose estimation performance compared to ViT-Pose. The bottom figure shows that `Argus` predicts keypoints that match better with the ground-truth, especially on the hip and elbow joints.

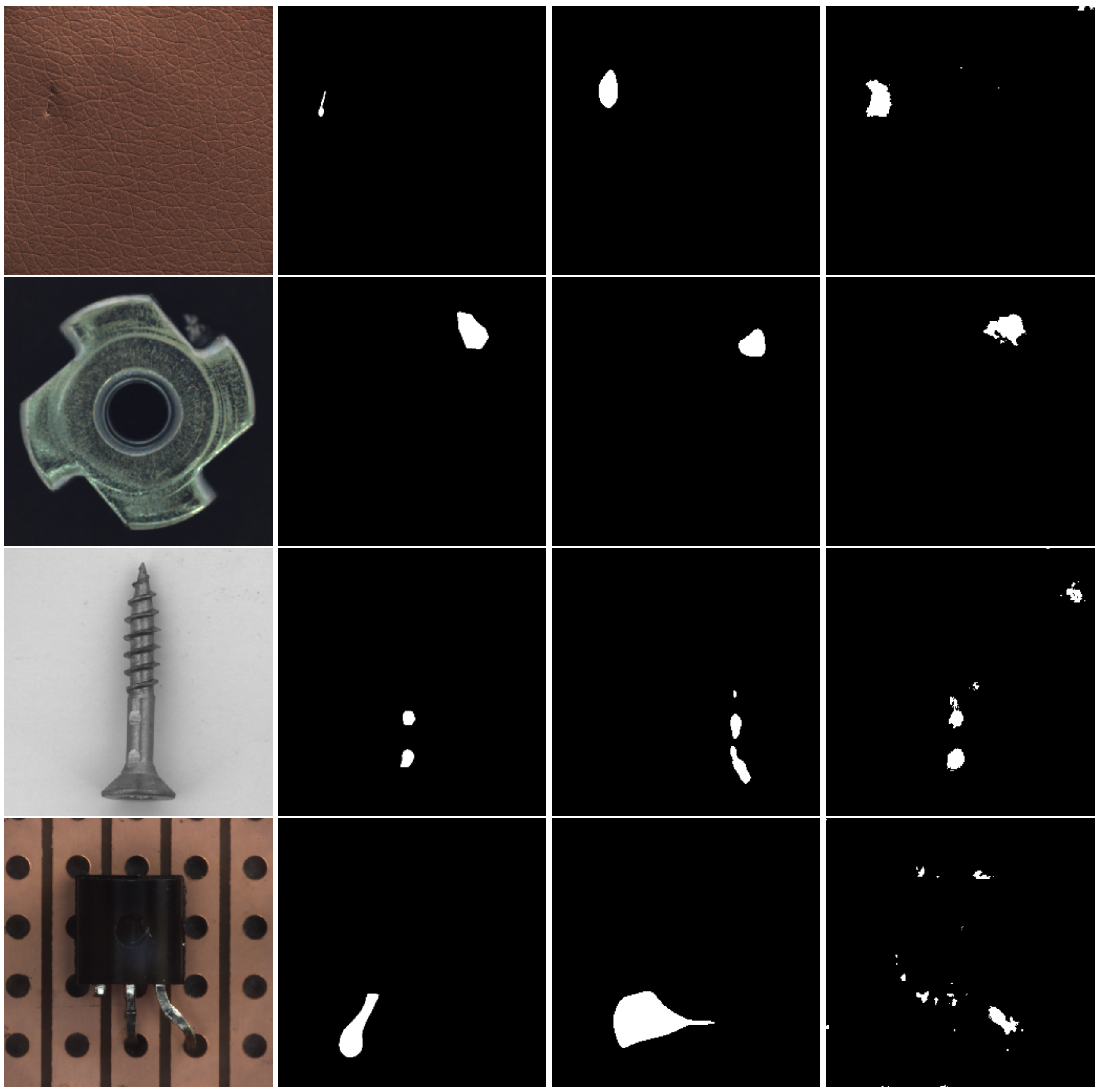

(a) Input image (b) Ground truth (c) `Argus` (d) DRAEM

Figure 12. Qualitative comparisons between `Argus` and DRAEM [86]. We visualize the results of these methods on MVTEC [2] test images. In comparison with DRAEM, `Argus` demonstrates strong performance in detecting anomalies with less noise in the predicted anomalous region, which is critical in deciding anomalies.

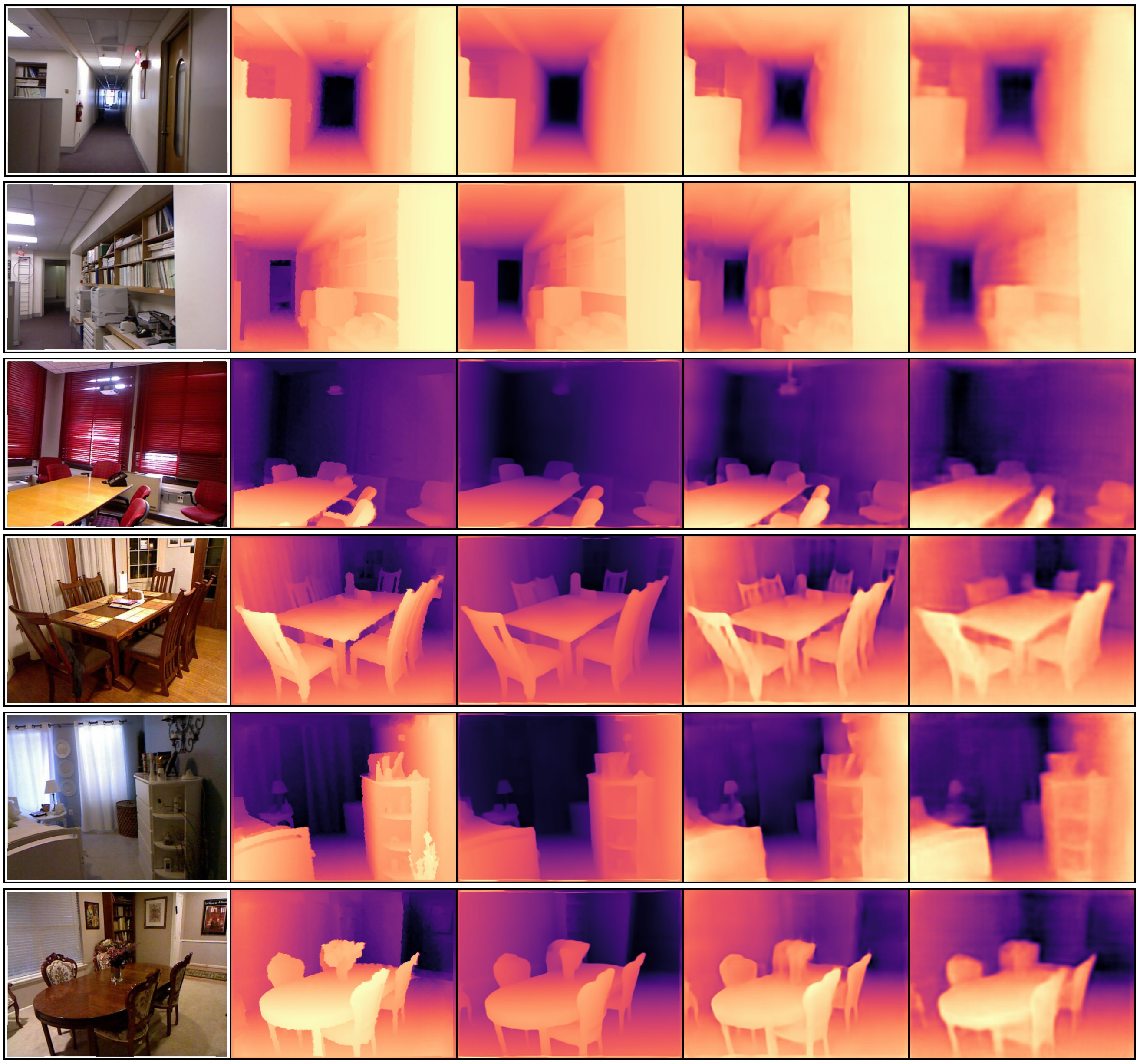


Figure 13. Qualitative comparisons between Argus, DINOv2 [57] and InvPT [79] on depth estimation task. We visualize the results of these methods on NYUv2 [65] test images. `Argus` outperforms DINOv2 and InvPT in fine-grained details and the prediction sharpness.

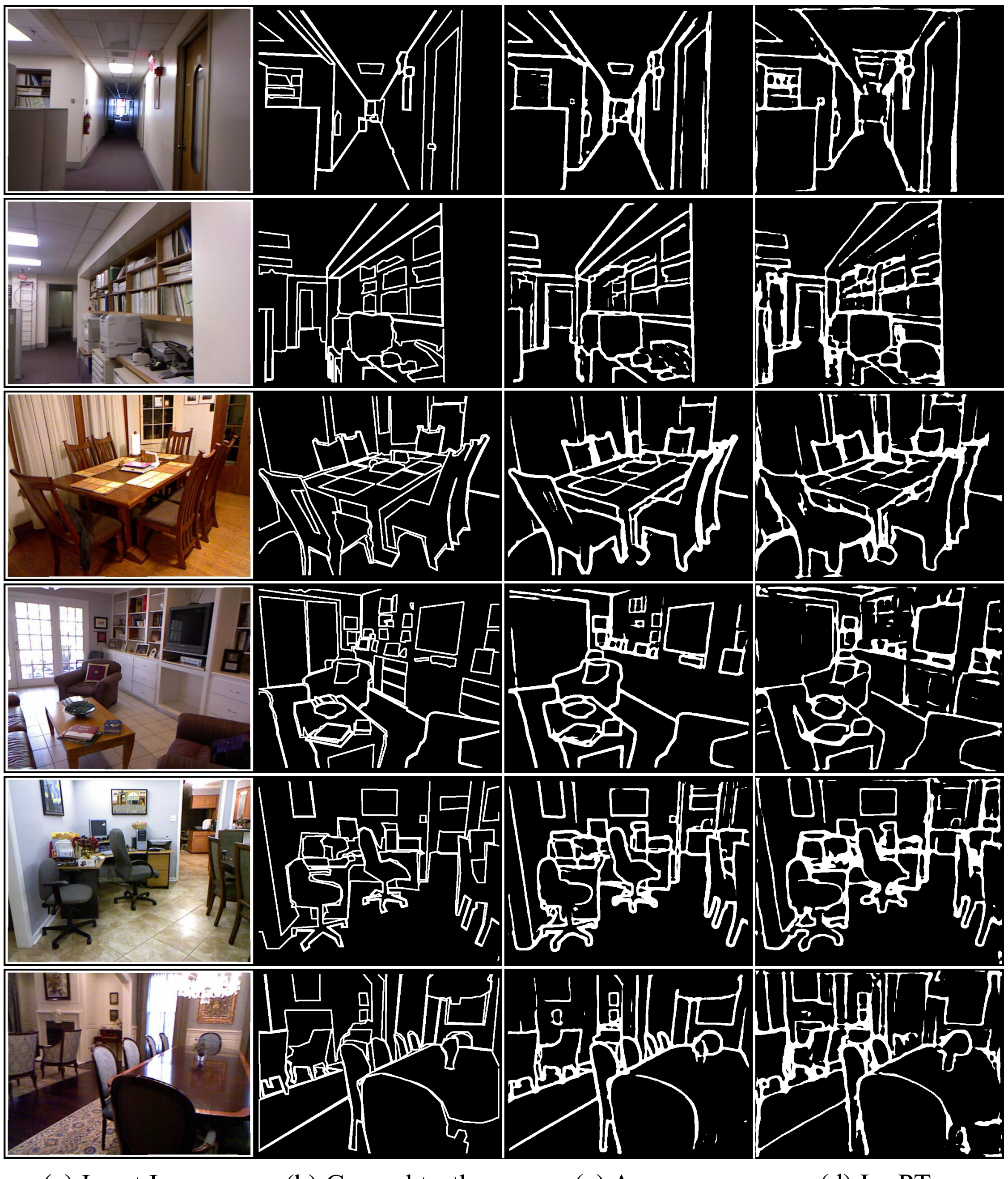


Figure 14. Qualitative comparisons between Argus and InvPT [79] on object boundary detection task. We visualize the results of these methods on NYUv2 [65] test images. `Argus` significantly outperforms InvPT and produces more accurate predictions.

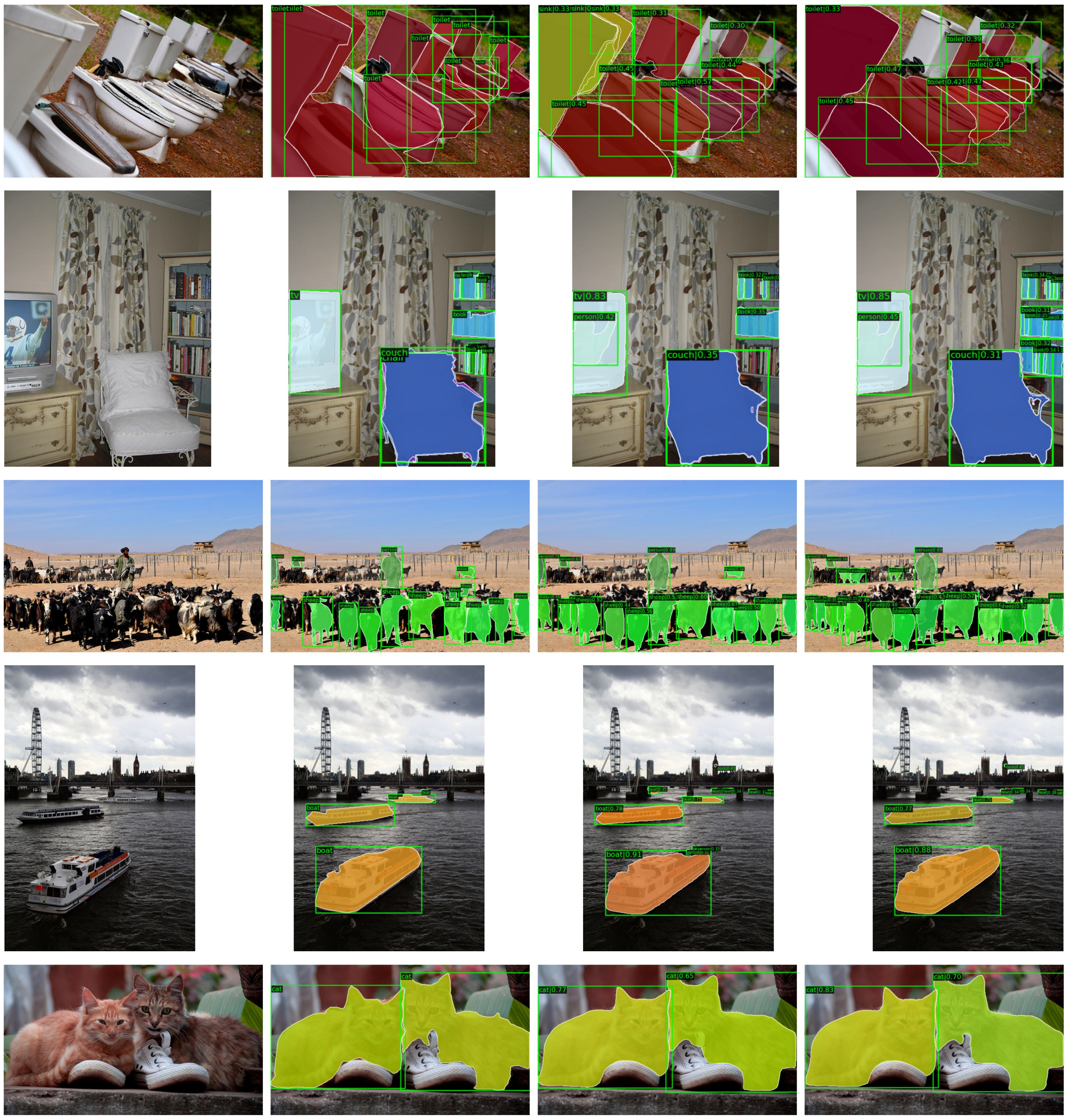


Figure 15. Qualitative comparisons between Argus and DINOv2 [57] on the instance segmentation task, which demonstrate higher accuracy of the Argus predictions. We use images from the test set of COCO dataset for this visualization.

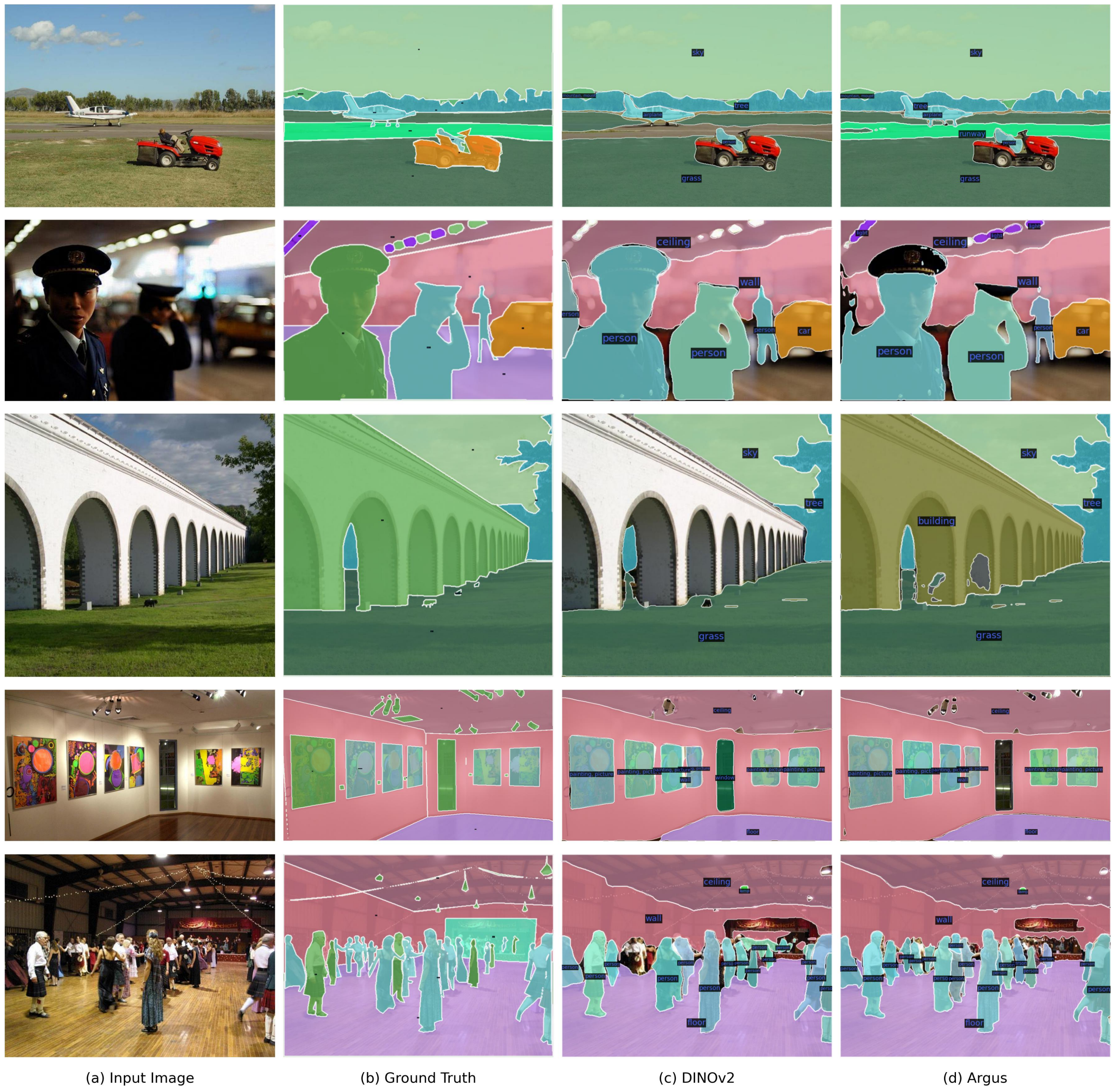


Figure 16. Qualitative comparisons between `Argus` and DINOv2 [57] on the panoptic segmentation task using ADE20K, which demonstrate higher accuracy of the `Argus` predictions. We use images from the test set of ADE20K dataset for this visualization.

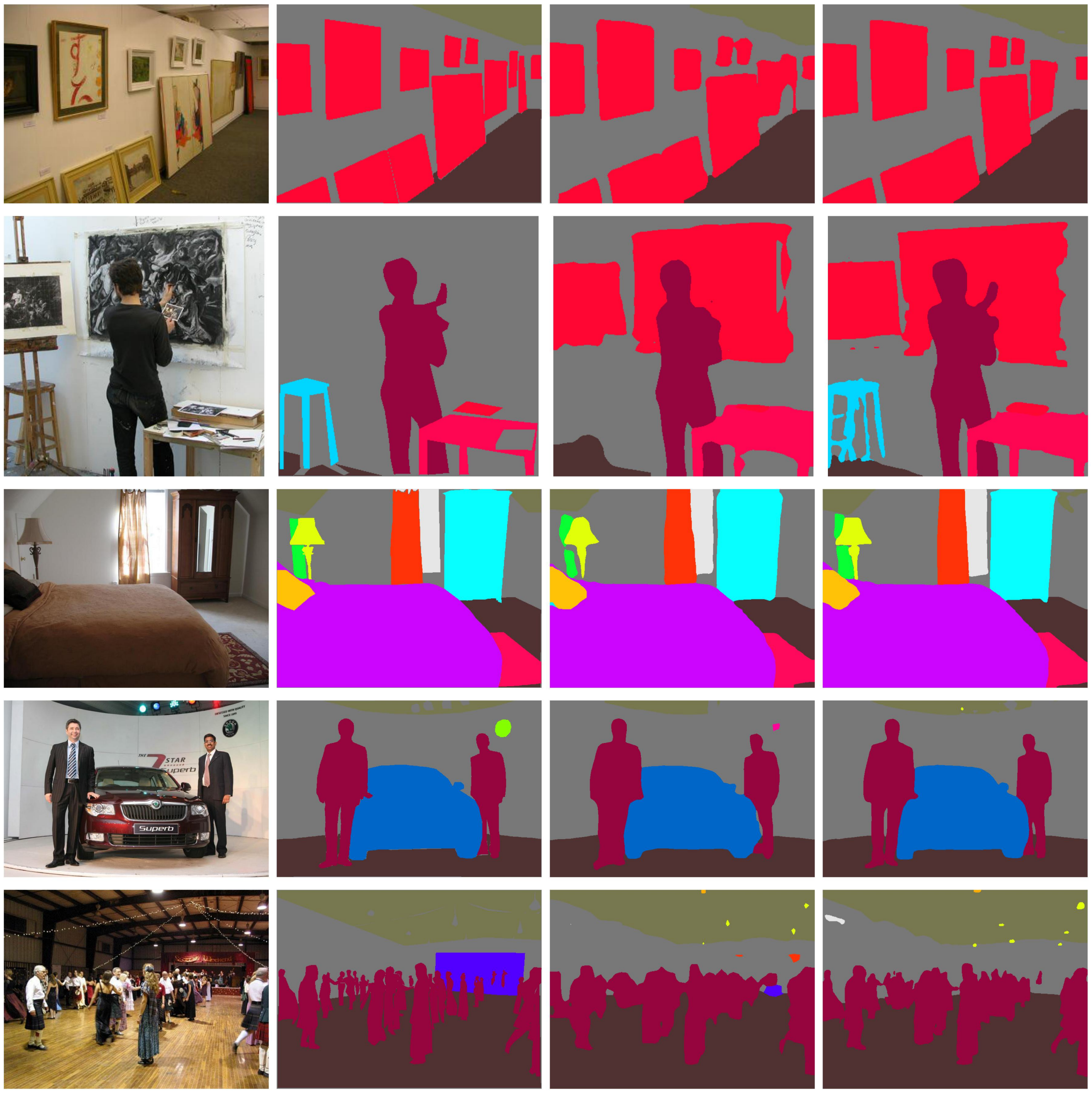


Figure 17. Qualitative comparisons between `Argus` and DINOv2 [57] on the semantic segmentation task. We use the ADE20K dataset for this visualization and observe that `Argus` improves over the segmentation maps generated from the baseline DINOv2 model.

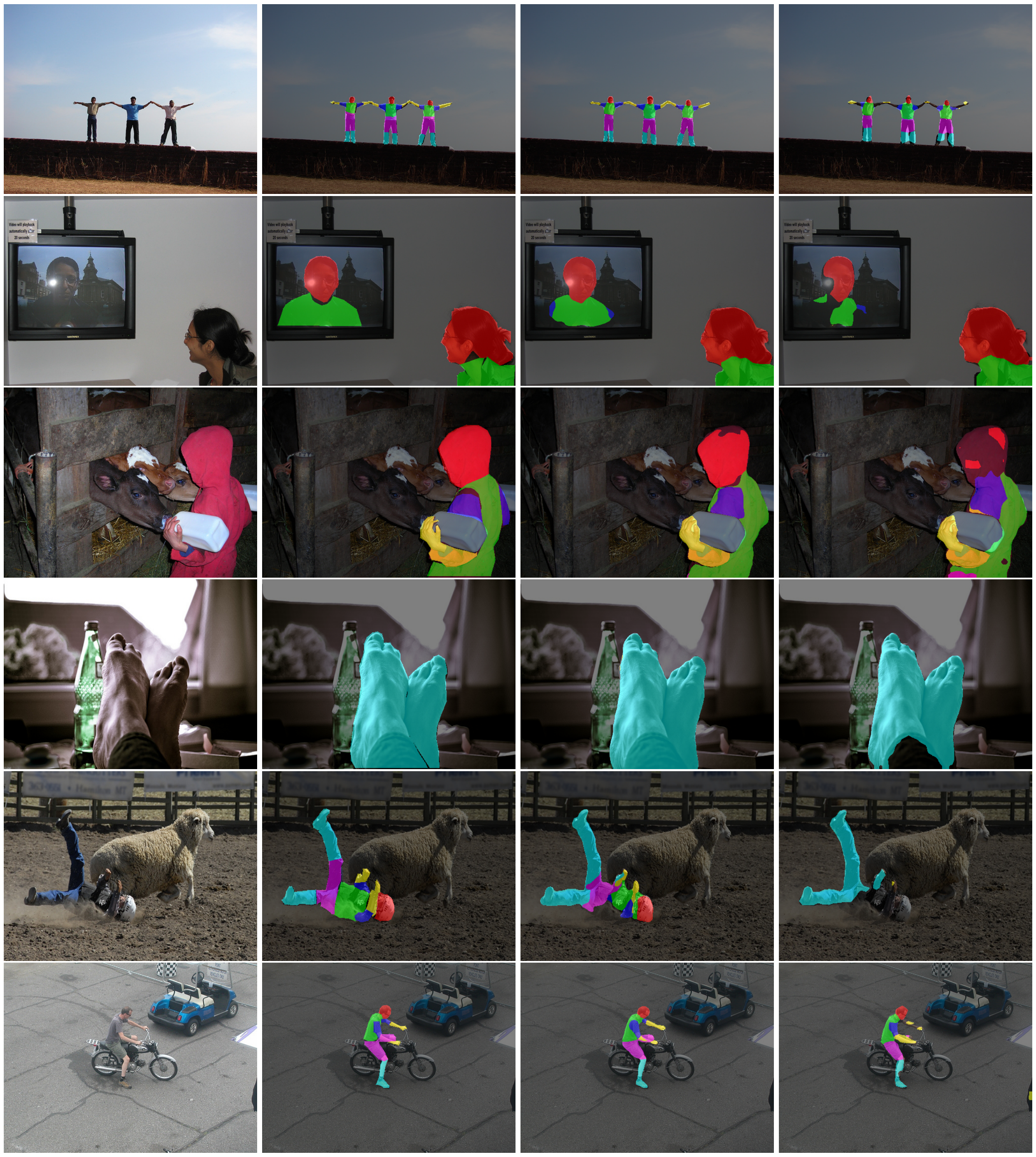

(a) Input Image (b) Ground-Truth (c) `Argus` (d) DINOv2

Figure 18. Qualitative comparisons between `Argus` and DINOv2 [57] on human parsing task using PASCAL-Context dataset [55]. `Argus` generated more accurate segmentation masks, especially in challenging cases.

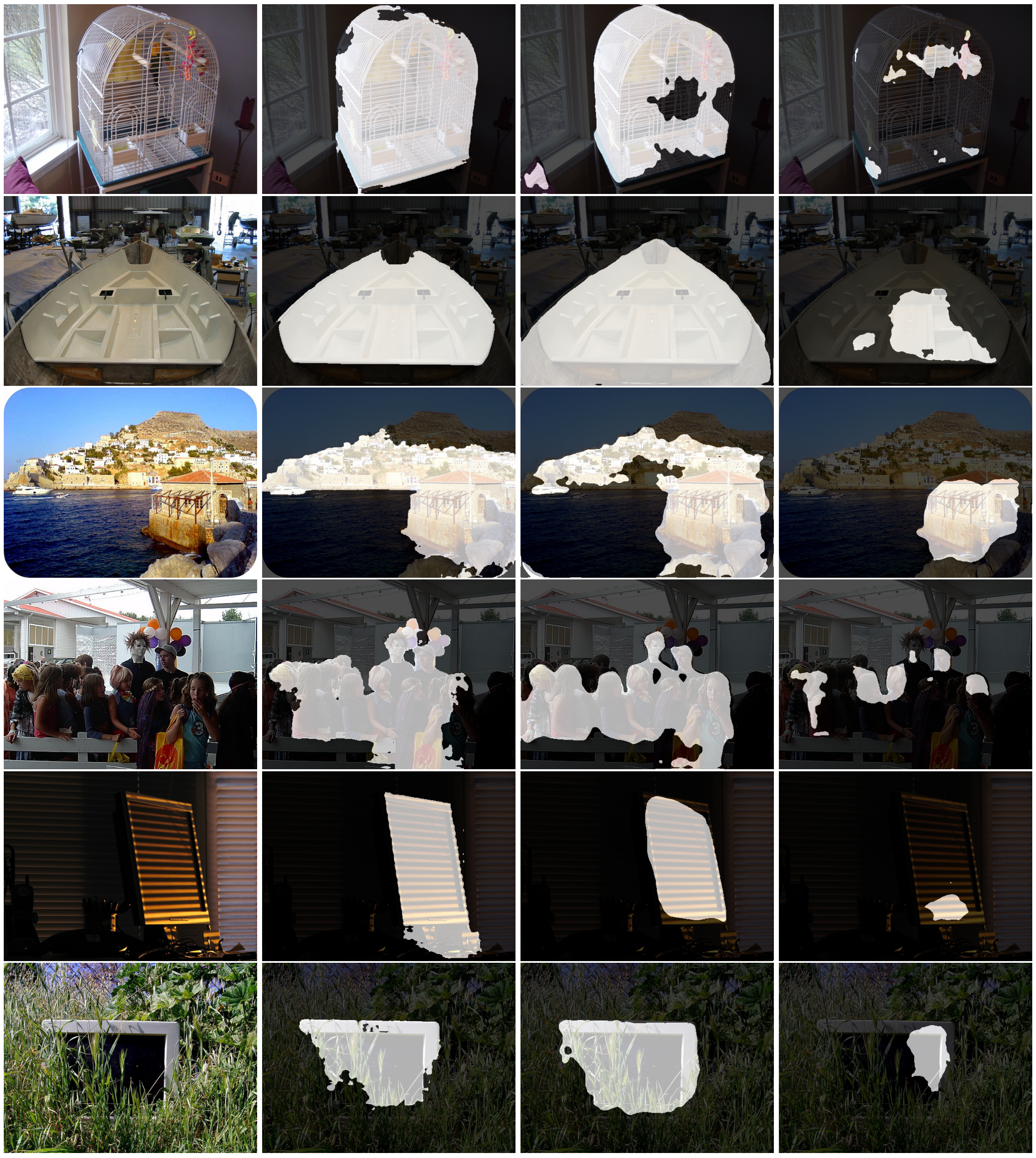


Figure 19. Qualitative comparisons between `Argus` and DINOv2 [57] on salient object detection task using PASCAL-Context [55] validation images. `Argus` outperforms DINOv2, producing more precise predictions.

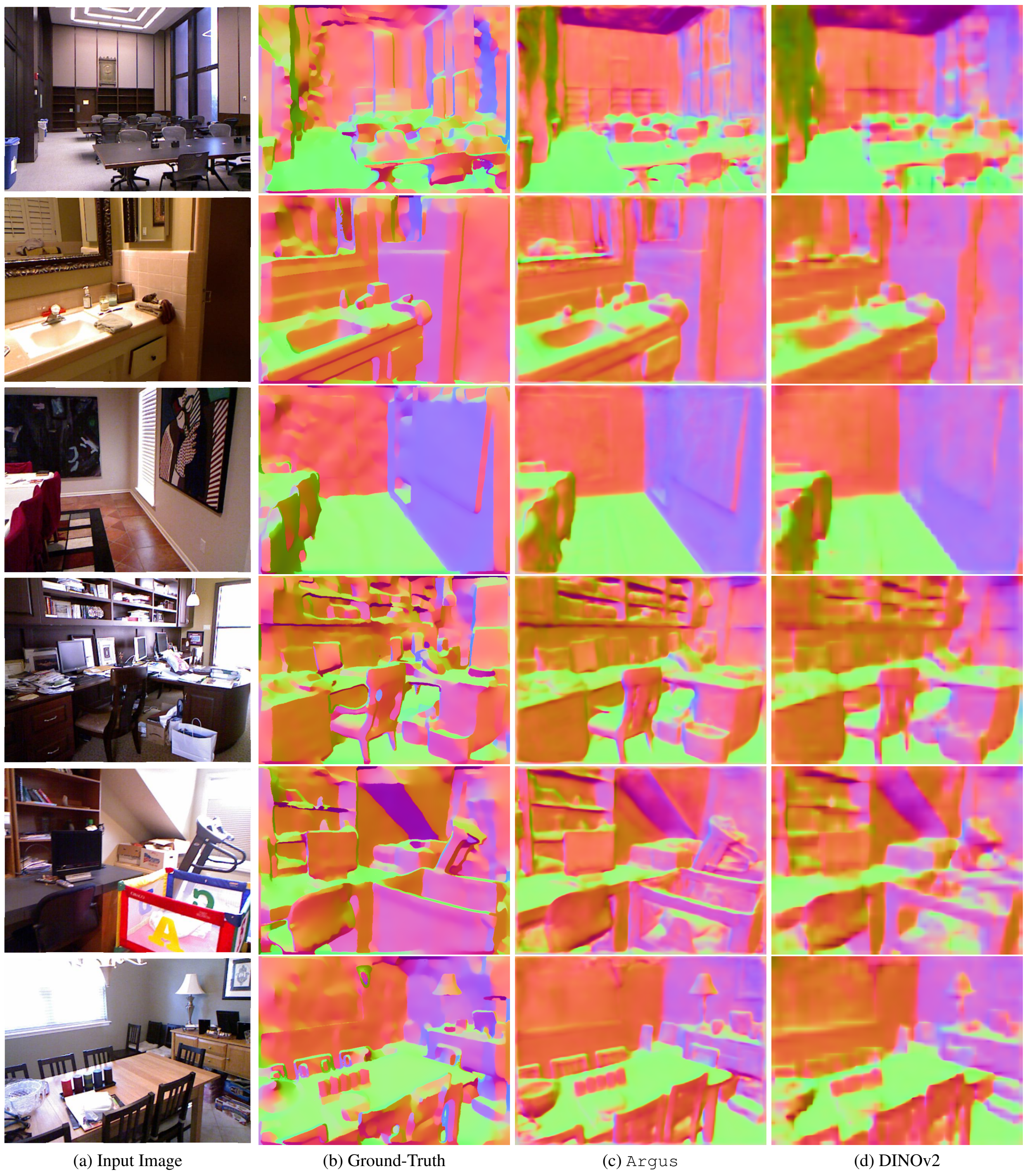


Figure 20. Qualitative comparisons of surface normals prediction between Argus and DINOv2 [57] on NYUv2 dataset [65]. Argus captures fine details in scenes with intricate structures, generating better surface normal maps compared to DINOv2.