Unifying Visual Perception by Dispersible Points Learning

Jianming Liang[1,2*], Guanglu Song[2], Biao Leng[1], and Yu Liu[2✉]

[1] School of Computer Science and Engineering, Beihang University
[2] SenseTime Research
ljmmm1997@gmail.com, songguanglu@sensetime.com,
lengbiao@buaa.edu.cn, liuyuisanai@gmail.com

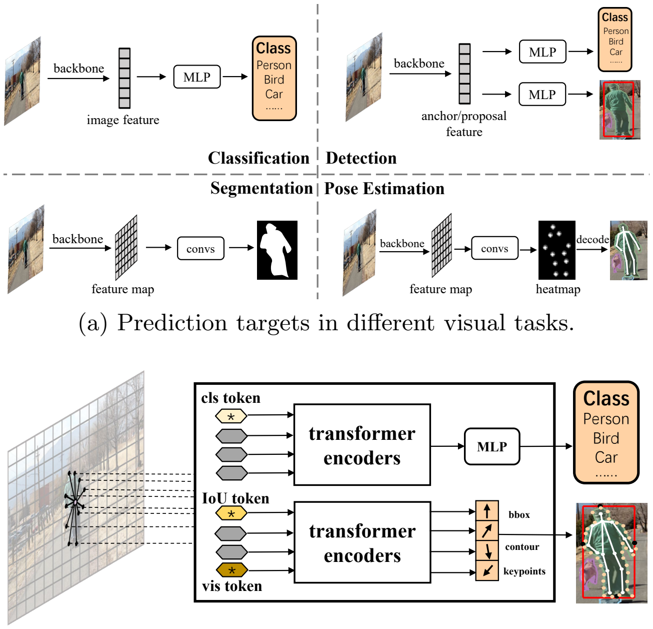
Abstract. We present a conceptually simple, flexible, and universal visual perception head for variant visual tasks, e.g., classification, object detection, instance segmentation and pose estimation, and different frameworks, such as one-stage or two-stage pipelines. Our approach effectively identifies an object in an image while simultaneously generating a high-quality bounding box or contour-based segmentation mask or set of keypoints. The method, called `UniHead`, views different visual perception tasks as the dispersible points learning via the transformer encoder architecture. Given a fixed spatial coordinate, `UniHead` adaptively scatters it to different spatial points and reasons about their relations by transformer encoder. It directly outputs the final set of predictions in the form of multiple points, allowing us to perform different visual tasks in different frameworks with the same head design. We show extensive evaluations on ImageNet classification and all three tracks of the COCO suite of challenges, including object detection, instance segmentation and pose estimation. Without bells and whistles, `UniHead` can unify these visual tasks via a single visual head design and achieve comparable performance compared to expert models developed for each task. We hope our simple and universal `UniHead` will serve as a solid baseline and help promote universal visual perception research. Code and models are available at https://github.com/Sense-X/UniHead.

Keywords: Dispersible points learning, transformer encoder, general visual perception

1 Introduction

Image classification [12], object detection [16,24], instance segmentation [24,8] and human pose estimation [24,1] are the vital visual perception tasks in computer vision. The vision community has rapidly improved results by developing robust feature representation. Regardless of the development of the powerful backbone, in large part, these advances are inseparable from the task-aware visual head structure design, such as TSD [36], CondInst [40] and CPN [7], or

[*] Work is done during the internship at SenseTime.



(a) Prediction targets in different visual tasks.

(b) Unifying visual perception by `UniHead`.

Fig. 1: (a). Illustration of the typical pipelines for different visual tasks. Different sub-tasks require different prediction targets and different feature structures. (b). Illustration of the `UniHead` design. Given a fixed spatial coordinate, `UniHead` adaptively scatters it to different spatial points and reasons about their relations by transformer encoders. It directly outputs a set of predictions in the form of multiple points to perform different visual tasks.

the elaborate frameworks construction, *e.g.*, one-stage detectors [23,26,41] and two-stage detectors [34,4]. These methods are conceptually experienced and introduce task exclusivity, *e.g.*, TSD [36] developed in object detection cannot be migrated to pose estimation. Our goal in this work is to develop a comparably generalized feature representation learning with task-agnostic structure design for *unifying visual perception*.

The main barriers behind this are: 1) As shown in Fig.1(a), the different prediction targets force the visual perception into different sub-tasks, *i.e.*, *a class label* for image classification, *a bounding box* for object detection, *a pixel-wised mask* for instance segmentation, and *a group of landmarks* for pose estimation. 2) How to conduct a task-agnostic head module which can generalize to all sub-tasks and frameworks while achieving good results? Given this, one might expect a complex head design is required to solve these barriers. However, we show that a surprisingly simple, flexible, and universal head module can easily generalize to different visual tasks or frameworks and surpass prior expert models in each individual task.

Our method, called UniHead, can be directly migrated to variant visual frameworks, *e.g.*, Faster RCNN [34], FCOS [41] and ATSS [50], by formulating the prediction targets as the dispersible points learning. As shown in Fig.1(b), UniHead is built upon any network backbone and the prediction targets for different tasks can be achieved by a basic yet effective points estimation. Given a fixed spatial coordinate, UniHead adaptively scatters it to different spatial points and reasons about their relations by several stacked transformer encoders. It directly outputs the final set of predictions in the form of multiple points, which is robust to geometric variations an object can exhibit, including scale, deformation, and orientation. For *image classification*, the points directly predict the object class. For *object detection*, the points are placed along the four edges of a bounding box. For *instance segmentation*, the points are evenly distributed along the instance mask contour. For *pose estimation*, the position of points conforms to the pose distribution of the training data.

Furthermore, we found it essential to adapt the initial position of the points according to different prediction targets. This can effectively alleviate the difficulty of optimization under the requirement of fitting objects with different scales and orientations. Additionally, the UniHead only adds a small computational overhead, enabling a universal system and rapid experimentation.

Without bells and whistles, UniHead can be equipped with popular backbones on different visual tasks, such as ResNet [18], ResNeXt [46], Swin Transformer [27], etc. It excels on the ImageNet [12] classification and all three tracks of the COCO [24] suite of challenges, including object detection, instance segmentation, and human pose estimation. We conduct extensive experiments to showcase the generality of our UniHead. By viewing each task as the dispersible points learning via the transformer encoder architecture, UniHead can perform comparably without any special design for specific tasks. UniHead, therefore, can be seen more broadly as a universal head module for visual perception and easily migrated to more complex tasks.

To summarize, our contributions are as follows:

1) We develop a comparably generalized dispersible points learning method for unifying visual perception. We hope our work can inspire the vision community to explore a unified vision framework.

2) We introduce the transformer encoder to reason about the relations of dispersible points and the adaptively points initialization to handle the geometric variations an object can exhibit, including scale, deformation, and orientation.

3) Detailed experiments on ImageNet [12] and MS-COCO [24] datasets show that UniHead can easily generalize to different tasks while obtaining comparable performance compared to the expert models developed in individual tasks.

2 Related Work

Image classification [12], object detection [16,24], instance segmentation [24,8] and pose estimation [1,24] are four popular tasks in computer vision. They all benefit a lot from the development of deep neural networks [18,37]. Among them, image classification [21] was the first to be applied with CNNs. The performance was improved by a considerable margin. After that, researchers are devoted to designing powerful backbones [18,46,19], which also give lift to other instance-level tasks, such as object detection [34,23] and human pose estimation [37].

For object detection, it requires bounding box level location and category information of interested instances in an image. The methods can be roughly categorized into three types: **Two-stage**, **One-stage** and **DETR** detectors. **Two-stage** methods detect a series of region proposals at first and refine them in the second stage. Faster RCNN [34] is a popular pipeline of the two-stage method, which also includes R-FCN [9], Cascade RCNN [4], Grid RCNN [29], *etc.*. **One-stage** methods predict locations and class scores on a large amount of pre-defined spatial candidates. They can be further divided into two types: anchor-based and anchor-free detectors. Anchor-based methods use anchor boxes as an initial set, such as SSD [26] and RetinaNet [23]. For anchor-free methods, some methods make dense predictions on spatial points, such as CenterNet (objects as points) [51], FCOS [41] and RepPoints [47]. And some other works obtain a keypoint heatmap first and get objects by grouping them. CornerNet [22], ExtremeNet [52] and CenterNet (keypoint triplets) [14] fall into this category. **DETR** methods, such as DETR [5], Deformable DETR [53] and Conditional DETR [30], propose to detect objects by decoding a pre-defined set of object queries with transformers. These queries are optimized one-to-one with ground truths so there is no need for NMS as post-processing. Such a way of one-to-one label assignment also inspires other works like Sparse RCNN [38].

For instance segmentation, it requires mask and class information for instances. The methods can be categorized into two types: **mask-based** and **contour-based**. **Mask-based** methods predict binary mask directly, which can further be divided into local-mask and global-mask methods. Most local-mask methods include two stages: the first one for instance detection and the second one for instance mask generation, such as Mask RCNN [17], PANet [25] and PointRend [20]. Global-mask methods usually predict the mask for the whole image and leverage dynamic mask filters to decode masks for different instances, such as YOLACT [3] and CondInst [40]. **Contour-based** methods obtain instance masks by predicting object boundaries. PolarMask [45] and Deep-Snake [31] are two typical works using this idea.

For human pose estimation, it requires the keypoint locations (*e.g.* nose, eyes, knees) for multiple humans in an image. There are mainly two kinds of approaches: **heatmap-based** and **regression-based**. **Heatmap-based** methods use a multi-class classifier to generate keypoint heatmaps and compose them with clustering and grouping procedures, such as CPN [7], HRNet [37] and DARK [49]. **Regression-based** methods, including Integral [39] and Center-Net [51], *etc.*, predict coordinates of keypoints directly. It is more simple to plug them into existing end-to-end learning frameworks.

Mask R-CNN [17], PointSetNet [44] and LSNet [15] achieved merging object detection, instance segmentation and pose estimation into one network. Besides these tasks, `UniHead` can be extended to image classification. Furthermore,

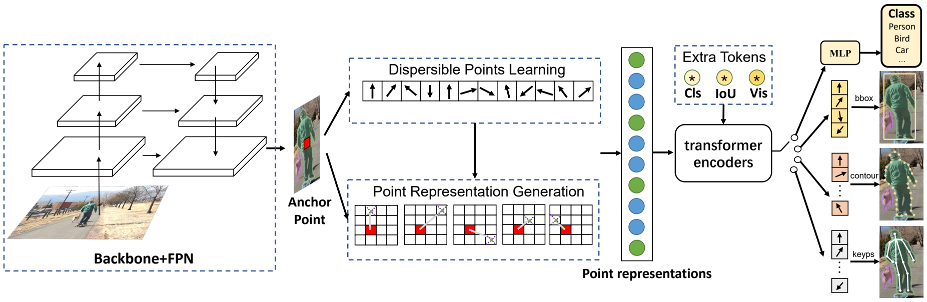


Fig. 2: A typical pipeline of UniHead. At first, most methods of location-sensitive tasks contain a backbone and the feature pyramid (not used in the image classification task) to extract feature maps. Then, for an anchor point, UniHead obtains multiple points via dispersible points learning. To generate point representations, bilinear interpolation is performed on the feature map according to point coordinates, which is denoted in dotted line. The obtained features will be concatenated with extra learnable tokens if necessary, and sent to corresponding transformer encoders to complete variant visual tasks.

UniHead can also be simply embedded in variant types of architectures, *e.g.*, anchor-free, anchor-based, and two-stage detectors, showing powerful ability on task and framework generalization.

3 Method

In this paper, we introduce the UniHead, a generalized visual head. It can be applied to different detection frameworks, such as Faster RCNN [34], FCOS [41] and ATSS [50], as well as different tasks including classification, object detection, instance segmentation and pose estimation. In this section, we first describe the design principle of UniHead and then detail the adaptation to different visual tasks and different visual frameworks. Finally, we delve into the inherent advantage of UniHead over other methods.

3.1 UniHead

In UniHead, given a fixed spatial coordinate $(A_x; A_y)$ (referred as **anchor point**), *i.e.*, *center point of a proposal or a point in the feature map*, it adaptively scatters it to different spatial points and reasons about the relations of them by several stacked transformer encoders. As shown in Fig.2, UniHead adopts the sequentially three-stage procedure to seek for the scattered point representations. In the first stage, it will generate the anchor representation $F_{x,y}$ according to the anchor coordinate or region proposal. For one-stage or anchor-free detectors, it is designated by the feature representation in the corresponding coordinate of the feature map. For the two-stage detectors, the feature generated by RoI Pooling [34] is used. In the second stage, K scattered points are generated by:

$$\begin{aligned} P_{x_i} &= A_x + s_x \quad x_i \\ P_{y_i} &= A_y + s_y \quad y_i; \end{aligned} \tag{1}$$

where $(\ x_i; \ \ y_i) = f(F_{x,y}; w_i)$. f is a simple multi-layer perceptron and w_i is the learnable parameter. $(s_x; s_y)$ is the computed scalar to modulate the magnitude of the $(\ x_i; \ \ y_i)$. Specifically, $(s_x; s_y)$ is the width and height of the region proposal in a two-stage detector, the anchor scale in a one-stage detector, and the model stride in an anchor-free detector. In the final stage, instead of quantizing a floating-number of $(P_{x_i}; P_{y_i})$, we perform bilinear interpolation to generate the point representations $F_{x_i, y_i}; i \ 2 \ [1; K]$.

To better reason about the relations of these scattered point representations and generate more informative features, we introduce the transformer operator to capture the correlative dependence between them. To improve the robustness of different visual tasks, we insert a task-aware token embedding by:

$$z_0 = [\mathbf{T}_{\textbf{task}}; F_{x_1,y_1}; F_{x_2,y_2}; :::; F_{x_K,y_K}]; \tag{2}$$

where $\mathbf{T}_{\textbf{task}}$ can be $\mathbf{T}_{\textbf{class}}$, $\mathbf{T}_{\textbf{IoU}}$, and $\mathbf{T}_{\textbf{visibility}}$ for image classification, object detection and pose estimation, respectively. The computation in transformer encoders for point representations can be formulated as:

$$\begin{aligned} & z^{\theta}_l = \mathrm{MHSA}(\mathrm{LN}(z_{l-1})) + z_{l-1}; \qquad l = 1 ::: L; \\ & z_l = \mathrm{MLP}(\mathrm{LN}(z^{\theta}_l)) + z^{\theta}_l; \qquad l = 1 ::: L; \\ & [\mathbf{T}^{\theta}_{\textbf{task}}; F^{\theta}_{x_1,y_1}; F^{\theta}_{x_2,y_2}; :::; F^{\theta}_{x_K,y_K}] = z_L; \end{aligned} \tag{3}$$

where MHSA means multi-head self attention in [43], LN indicates layer normalization [2], MLP is a multi-layer perceptron. Formally, during training, we use L transformer encoders, and the final output z_L will be adapted to different visual tasks to perform the task-aware prediction.

3.2 Adaptation to Different Visual Tasks

Image Classification. For image classification, we directly use the final feature map to perform dispersible points learning. The anchor point is set as the center of the input image and the corresponding scales are the input scale. We choose to align the classifier setting with standard vision transformers, *i.e.*, only leveraging classification token instead of all tokens in the classifier. The training can be formulated as:

$$L_{\mathrm{cls}} = \mathrm{CrossEntropy}(\mathrm{softmax}(\mathrm{MLP}(\mathbf{T}^{\theta}_{\textbf{cls}})); \mathrm{y}): \tag{4}$$

In the above y specifies the ground-truth class and MLP is a single fully-connected layer predicting the model's probability for the class with label y.

Object Detection. UniHead can be applied to a variety of detectors, such as Faster R-CNN [34], FCOS [41], *etc.*, without changing the backbone network structure, and the manner of label assignment. Specially, we concatenate a learnable token $\mathbf{T}_{\mathbf{IoU}}$ as a replacement for the IoU branch. After passing through all transformer blocks, the $\mathbf{T}'_{\mathbf{IoU}}$ is used to predict IoU, which will be multiplied by class prediction to get final scores at inference time. The F'_{x_i,y_i} is used to predict the offset for point (P_{x_i}, P_{y_i}). There are:

$$(P'_{x_i}, P'_{y_i}) = (P_{x_i}, P_{y_i}) + \mathrm{MLP}(F'_{x_i,y_i}) \odot (s_x, s_y), \tag{5}$$

where $\odot$ denotes element-wise multiplication, and the MLP is a single fully-connected layer shared between different points. The predicted bounding box can be computed by $B' = (\min\{P'_{x_i}\}, \min\{P'_{y_i}\}, \max\{P'_{x_i}\}, \max\{P'_{y_i}\})$, $i \in [1, K]$.

For the classification branch, it performs the same computational manner as UniHead in image classification. For regression, it shares z_0 with the classification branch to reduce the computational cost of point representation generation. Our loss function for detection is defined as:

$$L_{loc} = \frac{1}{n}\sum_{j=1}^{n} L_1(B'_j, B_j), \tag{6}$$

where j is the index of positive samples, B'_j is the predicted box and B_j is the ground truth. Other kinds of detection loss can also be used, *e.g.*, GIoU loss [35].

Instance Segmentation. For instance segmentation, we view this task as the contour-based regression. UniHead is placed at the output of the backbone to generate the points P'_{x_i,y_i} by Eq.1, Eq.2, Eq.3 and Eq.5. To align the point number between scattered points and the contour points in training data, we uniformly add new points, or delete points with the shortest edge until the target number is met, which is similar to Deep Snake [31]. All ground truth points are clockwise arranged around the contour line. The scattered points $\{P'_{x_i,y_i}, i \in [1, K]\}$ are uniformly and clockwisely perform one-to-one matching with them.

Besides, some objects are split into several components due to occlusions. To overcome this problem, we simply follow PolarMask [45] and directly treat them as multiple objects. During training, we use L_1 loss to optimize each point:

$$L_{seg} = \frac{1}{n}\sum_{i=1}^{n} L_1(P'_{x_i,y_i}, P_{x_i,y_i}), \tag{7}$$

where P'_{x_i,y_i} is the predicted point and P_{x_i,y_i} is the corresponding ground truth.

Pose Estimation The overall design of pose estimation is consistent with instance segmentation, except that an extra token $\mathbf{T}_{\mathbf{visibility}}$ is introduced to predict the visibility of keypoints. The number K of predicted points is aligned with keypoint number in the dataset. For pose estimation, each keypoint has a clear definition, like nose, eyes, *etc.*, which makes it possible to build one-to-one connection with dispersible points. l_1 loss is adopted to train the keypoint localization branch, same as Eq.7. For the training of keypoint visibility prediction, we use standard binary cross entropy loss.

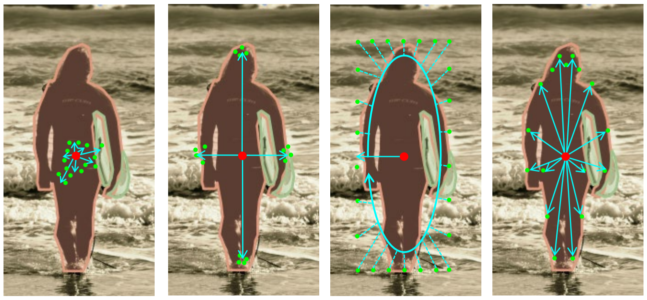


Fig. 3: Ways of point initialization for different tasks. From left to right: image classification, object detection, instance segmentation, pose estimation.

3.3 Adaptation to Different Visual Frameworks

Two-stage Framework. `UniHead` is applied to region proposals in the two-stage framework. Each proposal is represented as a combination of an anchor point (A_x, A_y) and its scale (s_x, s_y). The offsets $(\Delta x_i, \Delta y_i)$ are generated from the proposal feature extracted with RoI Pooling or RoI Align. Without other modifications, `UniHead` now can be directly used on a two-stage framework.

One-stage Framework. `UniHead` is applied on dense spatial points in the one-stage framework. For anchor-free methods, (A_x, A_y) and (s_x, s_y) are a point and the stride of the feature map. For anchor-based methods, (A_x, A_y) and (s_x, s_y) are the center point and the scale of an anchor. The offsets $(\Delta x_i, \Delta y_i)$ are generated using a 1×1 convolutional layer.

3.4 UniHead Initialization

To effectively alleviate the difficulty of optimization under the requirement of fitting objects with different scales and orientations, the result points are initialized in a more appropriate way for different tasks, which is illustrated in Fig.3. For image classification, points are casually scattered around the anchor point. For object detection, points are divided into four groups placed at the bottom, top, left, and right of the anchor point, respectively. For instance segmentation, first we set a 2D reference vector that starts from the anchor point. Based on the direction of this vector, the points are uniformly and clockwise initialized on the edge of a pseudo box generated from the anchor point and its spatial scale. For pose estimation, we calculate the average positions of different keypoints in the training dataset and use them to initialize points.

The initial point position is controlled by tuning the *bias* of the last fully-connected layer in MLP used for offsets generation. Taking object detection as an example, the *bias* for points at left, right, top and bottom are set to $[-0.5, 0]$, $[0.5, 0]$, $[0, -0.5]$ and $[0, 0.5]$, respectively.

Table 1: Ablation study on extra blocks for image classification task.

Method	GFLOPs	Top-1 acc.
ResNet-50	3.8	78.5
ResNet-50+extra blocks	4.2	79.0
ResNet-50+UniHead	4.1	**79.5**
Swin-T	4.5	81.2
Swin-T+extra blocks	5.2	81.7
Swin-T+UniHead	4.7	**81.8**
Swin-B	15.4	83.5
Swin-B+extra blocks	16.7	83.6
Swin-B+UniHead	15.7	**83.9**

Table 2: Ablation study on $\mathsf{T}_{\mathsf{task}}$. 'Det.' and 'Keyp.' mean detection and pose estimation, respectively.

Task	w/ $\mathbf{T}_{\mathbf{task}}$	AP	AP$_{.5}$	AP$_{.75}$	AP$_s$	AP$_m$	AP$_l$
Det.	x	41.6	**61.2**	**44.8**	23.4	45.1	56.2
	X	**41.8**	60.6	44.7	**23.8**	**45.1**	**56.7**
Keyp.	x	50.4	78.8	53.9	-	44.9	58.5
	X	**50.7**	**78.9**	**54.6**	-	**45.5**	**58.5**

4 Experiments

For image classification, experiments are conducted on the ILSVRC-2012 ImageNet [12] dataset with 1K classes and 1.3M images. We use Top-1 accuracy as the metric in classification experiments.

We also conduct experiments with different backbones on the MS-COCO 2017 [24] dataset, including object detection, instance segmentation, and human pose estimation tasks. For these tasks, training is performed on the *train* set, over 57K images for human pose estimation, and over 118K images for object detection and instance segmentation. For experiments of ablation studies, evaluation is conducted on the *val* set. We also report performance on the *test-dev* set to compare with the state-of-art methods. The mean average precision (AP) is used as the measurement in COCO experiments. But the definition of AP varies with tasks. For object detection and instance segmentation, AP is calculated under different IoU thresholds (bounding box IoU or mask IoU). For human pose estimation, AP is calculated with object keypoint similarity (OKS).

4.1 Implementation Details

In the image classification task, all models are trained using AdamW optimizer [28] with 1e-4 initial learning rate, 0.05 weight decay, $_1 = 0:9$, $_2 = 0:999$ and a batch size of 1024. We train classification models for 300 epochs and use consine annealing scheduler to decrease learning rate. Data augmentations in [42] are also used, *e.g.*, mix up, label smoothing, *etc.*.

For other three tasks, we use different backbones including ResNet [18], ResNeXt [46] and Swin Transformer [27] with weights pretrained on ImageNet [12]. For object detection, we use our UniHead on different detection pipelines and

Table 3: Ablation study on UniHead bias initialization strategy.

Task	UniHead Initialization?	AP	$AP_{.5}$	$AP_{.75}$
Det.	x	40.9	60.7	43.7
	X	**41.6**	**61.2**	**44.8**
Segm.	x	29.7	**53.5**	28.9
	X	**30.4**	53.2	**30.1**
Keyp.	x	57.0	81.9	62.4
	X	**57.9**	**82.6**	**63.9**

Table 4: Ablation study on point number. Point number 8, 16, 24, 32 are tried.

K	AP	$AP_{.5}$	$AP_{.75}$	AP_s	AP_m	AP_l
8	40.8	59.3	43.7	22.3	44.3	54.6
16	41.8	60.6	44.7	23.8	**45.1**	56.7
24	**41.8**	**60.7**	**44.7**	**23.9**	44.8	56.5
32	41.5	60.3	44.5	22.8	44.6	**56.8**

follow their original hyper-parameters. For instance segmentation and pose estimation, the same settings as Faster RCNN [34] are used. During training, we adopt AdamW [28] as the optimizer, with 1e-4 initial learning rate, 0.05 weight decay, $_1 = 0{:}9$ and $_2 = 0{:}999$. In our 1 setting, we train our model with mini-batch size 16 for 13 epochs and decrease the learning rate by a factor of 10 at epoch 9 and 12. Unless specified, the input scale of images is [800, 1333] and no data augmentations except horizontal flipping are used in training. The hyper-parameter of newly-added transformers keeps the same as [13].

4.2 Ablation Studies

In this section, we conduct extensive ablation studies on ImageNet and COCO *val* set to validate the effectiveness of UniHead on classification and localization tasks, respectively. Specially, for localization task, we choose object detection and all models are trained on Faster RCNN [34] baseline with AdamW optimizer [28] and ResNet-50 backbone for fair comparison. We find that AdamW can stably improve the performance by 1% AP compared to SGD.

Extra Blocks. We add extra blocks to the classification backbone networks to align their FLOPs with UniHead. Specifically, we append two bottlenecks to ResNet-50 ([3,4,6,5] for four stages) and two transformer blocks to Swin-T ([2,2,6,4] for four stages), whose results are shown in Table 1. Though additional layers can boost the performance, UniHead can achieve better performance with similar FLOPs. Also, we conduct the same experiment on Swin-B. We can see that when the model becomes bigger with higher FLOPs, extra blocks can hardly bring improvement. But UniHead achieves a continual performance boost. All these results prove that improvement brought by UniHead does not only account for its transformer blocks.

Task Token. We also explore the influence of $\mathbf{T}_{\mathbf{IoU}}$ and $\mathbf{T}_{\mathbf{visibility}}$ on object detection and pose estimation, respectively. As is shown in Table 2, the introduction of $\mathbf{T}_{\mathbf{task}}$ brings a slight improvement on both tasks, proving the effectiveness

Table 5: Ablation study on **block number**. L_{cls} and L_{loc} denote transformer encoder block number of classification and localization, respectively. #params means parameters of the detection head. The training and inference time is measured on a 16GB V100 GPU.

L_{cls}	L_{loc}	#params	GFLOPs	Training	Inference	AP	AP$_{.5}$	AP$_{.75}$
				(s/iter)	(ms/img)			
baseline		14.3M	215	0.38	82	38.8	59.9	42.1
1	1	11.9M	227	0.39	85	41.6	60.3	44.3
2	2	12.7M	239	0.40	87	41.6	60.4	44.7
3	3	13.5M	251	0.40	90	41.7	60.5	44.7
4	4	14.3M	263	0.41	92	**42.0**	**60.6**	**45.0**

Table 6: Ablation study on **different modules**. IoU prediction is not used in this table. "HD", "MHSA" and "DPL" mean head disentanglement, multi-head self attention and dispersible points learning, respectively.

HD	MHSA	DPL	AP	AP$_{.5}$	AP$_{.75}$	AP$_s$	AP$_m$	AP$_l$
x	x	x	38.8	59.9	42.1	22.1	41.9	51.9
X	x	x	39.3	60.0	42.5	22.0	42.9	52.6
X	X	x	39.9	60.5	43.4	22.4	43.2	53.4
X	x	X	40.7	**61.6**	44.4	23.1	43.4	55.1
X	X	X	**41.6**	61.2	**44.8**	**23.4**	**45.1**	**56.2**

of task tokens. It is worth noting that though visibility prediction is not used in pose estimation evaluation, $T_{visibility}$ still has a positive impact on training.

UniHead Initialization. We replace our task-specific bias initialization with zero initialization on different tasks. Main results are shown in Table 3. It proves that a proper initialization can help the unified architecture learn the knowledge of different tasks more quickly.

Point Number. We evaluate the performance of different point numbers in UniHead, which is shown in Table 4. It shows that our head can benefit from the increasing number of points. But more points may bring overfitting and more computational cost. So we choose to use $K = 16$ in our implementations.

Block Number. We also analyze the influence of the number of transformer encoder blocks. As is shown in Table 5, we compare the performances, head parameters, FLOPs, training time, and inference time with baseline under different block number settings. Our head can benefit slightly from the increase in block numbers. Considering computational costs and the head capacity, we finally use $L_{cls} = 2$ and $L_{loc} = 3$ in our implementations.

Head Disentanglement. To show that our method does not only benefit from the separated task heads, a Faster RCNN with sibling heads is given in the second row of Table 6. We simply remove the shared fully connected layers in the RCNN head and replace them with separated ones. We can observe that the improvement brought by head disentanglement (0.5 AP) is actually limited.

Dispersible Points Learning and Multi-head Self Attention. In order to demonstrate the effectiveness of dispersible points learning and multi-head self attention, we conduct experiments with different head designs and compare

Table 7: Results of UniHead with variant detection pipelines.

Method	AP	AP$_{.5}$	AP$_{.75}$
Faster RCNN [34]	38.8	59.9	42.1
+UniHead	**41.8**	**60.6**	**44.7**
Cascade RCNN [4]	42.1	60.8	45.3
+UniHead	**43.0**	**61.5**	**46.2**
ATSS (anchor-based) [50]	39.5	58.1	42.2
+UniHead	**40.6**	**58.3**	**44.2**
FCOS (w/o imprv.) [41]	37.1	56.3	39.1
+UniHead	**39.7**	**57.9**	**42.6**
Mask RCNN [17]	35.2	56.8	37.5
+UniHead (mask)	**37.0**	**57.9**	**39.9**

Table 8: Results of UniHead with variant backbones. "DCN" means deformable convolution. * means multi-scale training.

Method	Ours	AP	AP$_{.5}$	AP$_{.75}$
ResNet-50		38.8	59.9	42.1
ResNet-50	X	**41.8**	**60.6**	**44.7**
ResNet-101		39.9	60.5	43.5
ResNet-101	X	**42.4**	**61.4**	**45.7**
ResNeXt-101-64x4d		42.2	**63.4**	45.7
ResNeXt-101-64x4d	X	**44.5**	63.2	**48.0**
ResNeXt-101-64x4d-DCN		45.4	**67.1**	49.2
ResNeXt-101-64x4d-DCN	X	**47.3**	66.9	**51.3**
Swin-T*		43.7	66.4	47.7
Swin-T*	X	**46.3**	**66.4**	**49.5**

them with our head (without IoU prediction). First, we take the output of RoI Align [17] as tokens directly (49 in total), and process them with disentangled transformer encoders. The result is in the third row of Table 6. We can see that though more points are used, it still performs worse than DPL with $K = 16$.

Then, we leverage deformable RoI Pooling [10] as another form of dispersible points learning. Specifically, multiple offsets are generated in the same way and applied to deformable RoI Pooling for feature extraction. The result is shown in the fourth row of Table 6. It indicates that the combination of dispersible points learning and multi-head attention is more effective to capture semantic information within an instance.

4.3 Generalization Ability

Detection Pipeline Generalization. We evaluate the performance by transferring our UniHead to different detection pipelines. Specially, we simply replace the detection head in Mask RCNN with UniHead to build a mask-based version. As is shown in Table 7, the UniHead can boost the performance of all these types of detectors, showing its generalization ability on different detection frameworks.

Backbone Generalization. We further conduct experiments with different backbones under the setting of Faster RCNN. As is shown in Table 8, our head can steadily boost the performance by 2 3% AP. It demonstrates the generalization ability of our method on variant backbones.

Table 9: Results on different tasks. "*" indicates multi-scale training, multi-stage refinement and 11x scheduler. "+" is multi-scale training and 2x scheduler.

Task	Method	backbone	Top-1 acc.	AP	$\text{AP}_{.5}$	$\text{AP}_{.75}$
Cls.	baseline	R50	78.5	-	-	-
	UniHead		**79.5**	-	-	-
Det.	Faster RCNN	R50	-	38.8	59.9	42.1
	UniHead		-	**41.8**	**60.6**	**44.7**
	Mask RCNN		-	39.0	59.8	42.4
	UniHead (box)		-	**42.3**	**60.9**	**45.5**
Segm.	DeepSnake [31]	DLA34	-	30.3	-	-
	UniHead	R50	-	**30.4**	53.2	30.1
Keyp.	PointSet* [44]	R50	-	**58.0**	80.8	62.4
	UniHead^{+}		-	57.9	**82.6**	**63.9**

Task Generalization. As mentioned before, our head is a unifying perception head, which means that it can be applied to variant visual tasks. To be specific, we use $K = 16$ for image classification and object detection, $K = 36$ for instance segmentation and $K = 17$ points for human pose estimation. The baseline of classification is trained with the same setting as UniHead for fair comparison. The performance is evaluated on ImageNet *val* set for classification, and COCO *val* set for other three tasks. The experimental results are shown in Table 9. We can see that with a ResNet-50 backbone, the UniHead makes improvements on classification and object detection, and get a close performance compared with expert models for instance segmentation and pose estimation.

4.4 Comparison with State-of-the-Art

We evaluate object detection, instance segmentation and pose estimation on COCO *test-dev*, whose results are shown in Table 10. The reported AP is related to corresponding tasks, *e.g.*, mask AP for instance segmentation. We only adopt multi-scale training for data augmentation and no TTA is used. **It should be noted that we don't introduce any task-aware algorithm design, *e.g.*, multi-stage refinement for pose estimation.**

For object detection, the experimental setting in multi-scale training is $[480, 960]$ for image minimum side and 1333 for image maximum side. We can see that with stronger backbones, our UniHead can achieve competitive performance, although it is not developed just for object detection. For instance segmentation, the same augmentation strategy as object detection is used. Here we also use the mask head of Mask RCNN [17] to build a mask-based UniHead. Without bells and whistles, UniHead gets 46.7% AP with mask-based head and 39.4% AP with contour-based head. Compared with expert models, UniHead achieves comparable performance only using a simpler pipeline. For pose estimation, we use a larger resolution of input image ($[480, 1200]$ for image minimum side and 2000 for image maximum side). With a surprisingly simple way, *i.e.*, direct keypoint regression using l_1 loss, UniHead gets a close performance compared with other regression-based methods which utilize multi-stage refinement (like [44]) and more iterations of training.

Table 10: Comparisons of for different algorithms and different tasks evaluated on the COCO *test-dev* set. "FG" and "TG" indicate that the method can be generalized to different visual frameworks and visual tasks, respectively. "*" denotes multi-scale test.

Method	FG	TG	backbone	iteration	AP	AP$_{.5}$	AP$_{.75}$	AP$_S$	AP$_M$	AP$_L$
Object Detection										
ATSS [50]	x	x	X-101-64x4d-DCN	2x	47.7	65.5	51.9	29.7	50.8	59.4
BorderDet [33]	x	x	X-101-64x4d-DCN	2x	48.0	67.1	52.1	29.4	50.7	60.5
Deformable DETR [53]	x	x	X-101-64x4d-DCN	∼4x	50.1	69.7	54.6	30.6	52.8	64.7
DynamicHead [11]	X	x	X-101-64x4d-DCN	2x	52.3	70.7	57.2	35.1	56.2	63.4
PointSet [44]	x	X	X-101-64x4d-DCN	2x	45.1	66.1	48.9	-	-	-
LSNet [15]	x	X	X-101-64x4d-DCN	2x	49.6	69.0	54.1	30.3	52.8	62.8
UniHead	X	X	X-101-64x4d-DCN	2x	50.5	70.0	54.4	31.2	53.4	64.7
UniHead	X	X	Swin-L	2x	**54.7**	**74.5**	**59.1**	**35.6**	**58.2**	**70.2**
Instance Segmentation										
Mask-based:										
Mask RCNN [17]	x	X	X-101-32x4d	1x	37.1	60.0	39.4	16.9	39.9	53.5
HTC [6]	x	X	X-101-64x4d	∼2x	41.2	63.9	44.7	22.8	43.9	54.6
YOLACT [3]	x	x	ResNet-101	4x	31.2	50.6	32.8	12.1	33.3	47.1
DetectoRS [32]	x	x	X-101-32x4d	3x	45.8	69.2	50.1	27.4	48.7	59.6
UniHead (w/ mask head)	X	X	X-101-64x4d-DCN	3x	43.6	67.1	47.0	25.1	46.5	58.1
UniHead (w/ mask head)	X	X	Swin-L	3x	**46.7**	**71.2**	**50.8**	**28.2**	**50.3**	**62.1**
Contour-based:										
ExtremeNet [52]	x	X	HG-2 stacked	∼8x	18.9	44.5	13.7	10.4	20.4	28.3
DeepSnake [31]	x	x	DLA-34 [48]	∼11x	30.3	-	-	-	-	-
PolarMask [45]	x	x	X-101-64x4d-DCN	2x	36.2	59.4	37.7	17.8	37.7	51.5
PointSet [44]	x	X	X-101-64x4d-DCN	2x	34.6	60.1	34.9	45.1	66.1	48.9
LSNet [15]	x	X	X-101-64x4d-DCN	∼2x	37.6	64.0	38.3	22.1	39.9	49.1
UniHead	X	X	X-101-64x4d-DCN	2x	36.6	63.0	36.2	22.0	38.6	48.5
UniHead	X	X	Swin-L	2x	**39.4**	**67.0**	**39.3**	**24.7**	**41.7**	**52.0**
Pose Estimation										
Heatmap-based:										
CPN [7]	x	x	ResNet-Inception	-	72.1	91.4	80.0	-	68.7	77.2
HRNet [37]	x	x	HRNet-W48	∼16x	75.5	92.5	83.3	-	71.9	81.5
DARK [49]	x	x	HRNet-W48	∼11x	**76.2**	**92.5**	**83.6**	-	**72.5**	**82.4**
Regression-based:										
CenterNet* [51]	x	X	HG-2 stacked	∼11x	63.0	86.8	69.6	-	58.9	70.4
PointSet [44]	x	X	X-101-64x4d-DCN	∼8x	62.5	83.1	68.3	-	-	-
LSNet [15]	x	X	X-101-64x4d-DCN	∼6x	59.0	83.6	65.2	-	53.3	67.9
UniHead	X	X	X-101-64x4d-DCN	2x	65.4	87.3	72.6	-	60.9	72.3
UniHead	X	X	Swin-L	2x	**66.1**	**88.7**	**73.7**	-	**62.0**	**72.3**

5 Conclusion

In this paper, we proposed `UniHead`, a unifying visual perception head. It can not only be embedded in variant detection frameworks, but also applied to different visual tasks, including image classification, object detection, instance segmentation and pose estimation. `UniHead` perceives instances by dispersible points learning, which is also equipped with transformer encoders to capture semantic relations of them. Though our `UniHead` is designed in a simple way, it achieves comparable performance on each task compared with expert models. This work shows the potential in general visual learning and we hope it can promote universal visual perception research.

Acknowledgement: The work was supported by the National Key R&D Program of China under Grant 2019YFB2102400.

References

1. Andriluka, M., Pishchulin, L., Gehler, P., Schiele, B.: 2d human pose estimation: New benchmark and state of the art analysis. In: CVPR. pp. 3686–3693 (2014)
2. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv:1607.06450 (2016)
3. Bolya, D., Zhou, C., Xiao, F., Lee, Y.J.: Yolact: Real-time instance segmentation. In: ICCV. pp. 9157–9166 (2019)
4. Cai, Z., Vasconcelos, N.: Cascade r-cnn: Delving into high quality object detection. In: CVPR. pp. 6154–6162 (2018)
5. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: ECCV. pp. 213–229 (2020)
6. Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Shi, J., Ouyang, W., et al.: Hybrid task cascade for instance segmentation. In: CVPR. pp. 4974–4983 (2019)
7. Chen, Y., Wang, Z., Peng, Y., Zhang, Z., Yu, G., Sun, J.: Cascaded pyramid network for multi-person pose estimation. In: CVPR. pp. 7103–7112 (2018)
8. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: CVPR. pp. 3213–3223 (2016)
9. Dai, J., Li, Y., He, K., Sun, J.: R-fcn: Object detection via region-based fully convolutional networks. NeurIPS **29** (2016)
10. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: ICCV. pp. 764–773 (2017)
11. Dai, X., Chen, Y., Xiao, B., Chen, D., Liu, M., Yuan, L., Zhang, L.: Dynamic head: Unifying object detection heads with attentions. In: CVPR. pp. 7373–7382 (2021)
12. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR. pp. 248–255 (2009)
13. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv:2010.11929 (2020)
14. Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., Tian, Q.: Centernet: Keypoint triplets for object detection. In: ICCV. pp. 6569–6578 (2019)
15. Duan, K., Xie, L., Qi, H., Bai, S., Huang, Q., Tian, Q.: Location-sensitive visual recognition with cross-iou loss. arXiv:2104.04899 (2021)
16. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. IJCV **88**(2), 303–338 (2010)
17. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV. pp. 2961–2969 (2017)
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
19. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: CVPR. pp. 4700–4708 (2017)
20. Kirillov, A., Wu, Y., He, K., Girshick, R.: Pointrend: Image segmentation as rendering. In: CVPR. pp. 9799–9808 (2020)
21. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. NeurIPS **25** (2012)
22. Law, H., Deng, J.: Cornernet: Detecting objects as paired keypoints. In: ECCV. pp. 734–750 (2018)
23. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV. pp. 2980–2988 (2017)

24. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV. pp. 740–755 (2014)
25. Liu, S., Qi, L., Qin, H., Shi, J., Jia, J.: Path aggregation network for instance segmentation. In: CVPR. pp. 8759–8768 (2018)
26. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: ECCV. pp. 21–37 (2016)
27. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. arXiv:2103.14030 (2021)
28. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv:1711.05101 (2017)
29. Lu, X., Li, B., Yue, Y., Li, Q., Yan, J.: Grid r-cnn. In: CVPR. pp. 7363–7372 (2019)
30. Meng, D., Chen, X., Fan, Z., Zeng, G., Li, H., Yuan, Y., Sun, L., Wang, J.: Conditional detr for fast training convergence. In: ICCV. pp. 3651–3660 (2021)
31. Peng, S., Jiang, W., Pi, H., Li, X., Bao, H., Zhou, X.: Deep snake for real-time instance segmentation. In: CVPR. pp. 8533–8542 (2020)
32. Qiao, S., Chen, L.C., Yuille, A.: Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. In: CVPR. pp. 10213–10224 (2021)
33. Qiu, H., Ma, Y., Li, Z., Liu, S., Sun, J.: Borderdet: Border feature for dense object detection. In: ECCV. pp. 549–564 (2020)
34. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. NeurIPS (2015)
35. Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.: Generalized intersection over union: A metric and a loss for bounding box regression. In: CVPR. pp. 658–666 (2019)
36. Song, G., Liu, Y., Wang, X.: Revisiting the sibling head in object detector. In: CVPR. pp. 11563–11572 (2020)
37. Sun, K., Xiao, B., Liu, D., Wang, J.: Deep high-resolution representation learning for human pose estimation. In: CVPR. pp. 5693–5703 (2019)
38. Sun, P., Zhang, R., Jiang, Y., Kong, T., Xu, C., Zhan, W., Tomizuka, M., Li, L., Yuan, Z., Wang, C., et al.: Sparse r-cnn: End-to-end object detection with learnable proposals. In: CVPR. pp. 14454–14463 (2021)
39. Sun, X., Xiao, B., Wei, F., Liang, S., Wei, Y.: Integral human pose regression. In: ECCV. pp. 529–545 (2018)
40. Tian, Z., Shen, C., Chen, H.: Conditional convolutions for instance segmentation. In: ECCV. pp. 282–298 (2020)
41. Tian, Z., Shen, C., Chen, H., He, T.: Fcos: Fully convolutional one-stage object detection. In: ICCV. pp. 9627–9636 (2019)
42. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: ICML. pp. 10347–10357 (2021)
43. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. NeurIPS **30** (2017)
44. Wei, F., Sun, X., Li, H., Wang, J., Lin, S.: Point-set anchors for object detection, instance segmentation and pose estimation. In: ECCV. pp. 527–544 (2020)
45. Xie, E., Sun, P., Song, X., Wang, W., Liu, X., Liang, D., Shen, C., Luo, P.: Polarmask: Single shot instance segmentation with polar representation. In: CVPR. pp. 12193–12202 (2020)
46. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: CVPR. pp. 1492–1500 (2017)

47. Yang, Z., Liu, S., Hu, H., Wang, L., Lin, S.: Reppoints: Point set representation for object detection. In: ICCV. pp. 9657–9666 (2019)
48. Yu, F., Wang, D., Shelhamer, E., Darrell, T.: Deep layer aggregation. In: CVPR. pp. 2403–2412 (2018)
49. Zhang, F., Zhu, X., Dai, H., Ye, M., Zhu, C.: Distribution-aware coordinate representation for human pose estimation. In: CVPR. pp. 7093–7102 (2020)
50. Zhang, S., Chi, C., Yao, Y., Lei, Z., Li, S.Z.: Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In: CVPR. pp. 9759–9768 (2020)
51. Zhou, X., Wang, D., Krähenbühl, P.: Objects as points. arXiv:1904.07850 (2019)
52. Zhou, X., Zhuo, J., Krahenbuhl, P.: Bottom-up object detection by grouping extreme and center points. In: CVPR. pp. 850–859 (2019)
53. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. arXiv:2010.04159 (2020)