PolarMask: Single Shot Instance Segmentation with Polar Representation （CVPR2020）

Enze Xie[1,2]*, Peize Sun[3]*, Xiaoge Song[4]*, Wenhai Wang[4], Ding Liang[2], Chunhua Shen[5], Ping Luo[1]

1The University of Hong Kong

2Sensetime Group Ltd

3Xi’an Jiaotong University

4Nanjing University 5The University of Adelaide

动机

- 在instance segmentation领域，主流的是以Makrcnn为代表的两阶段算法，且主流的改进主要集中在对检测核心的改进
  - 检测目标框→分割目标
- Instance segmentation需要给出point-to-point的输出，这个是不经济的
- 可以将分割看做边缘的标定，如果边缘点足够多；利用polar representation可以更好表示边缘

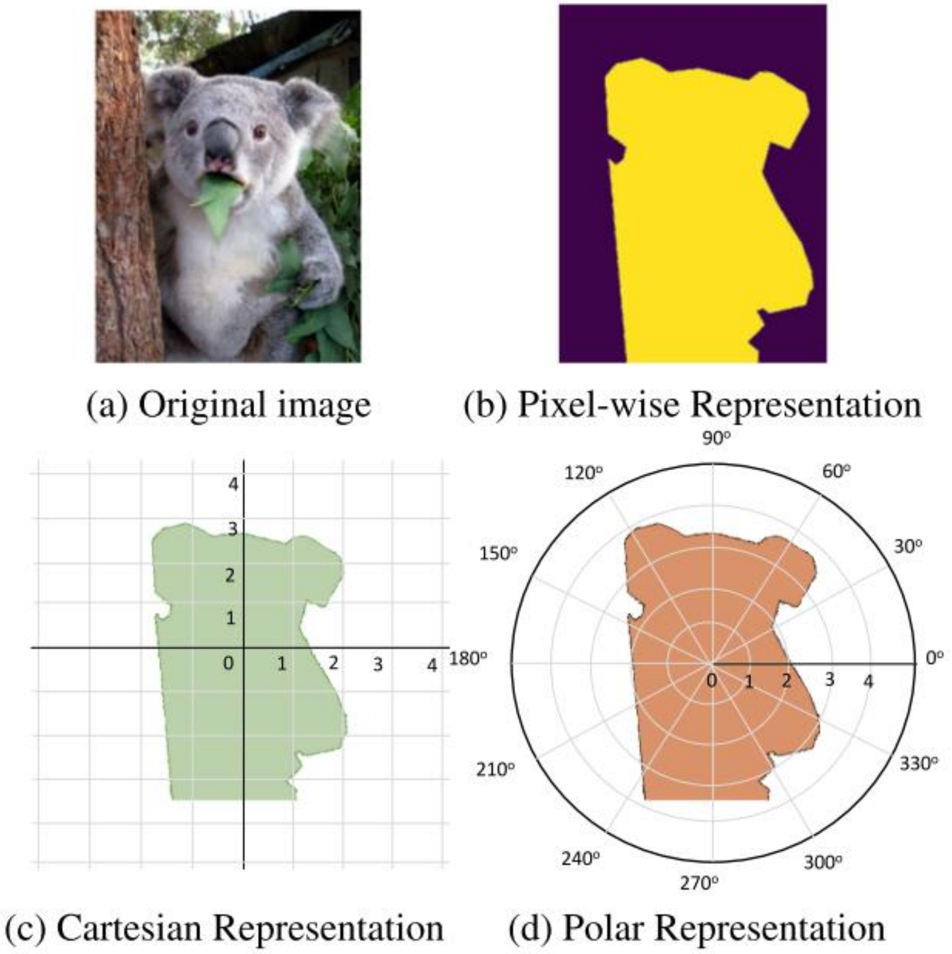


Figure 1 – Instance segmentation with different mask representations. (a) is the original image. (b) is the pixel-wise mask representation. (c) and (d) represent a mask by its contour, in the Cartesian and Polar coordinates, respectively.

方案思路

- 边缘表示:
  - 首先定义物体中心作为原点
  - 一段边缘可以定义为（角度，距离）
  - 极坐标表达（角度，距离）更自然
- Instance segmentation=center detection + contour regression
- 设计了一种即插即用的模块，可以接入任一检测网络，将其改成instance segmentation网络
- 和检测网络相比，几乎不增加网络的负担就实现了分割
- 提出PolarIOU loss来训练这一网络模块

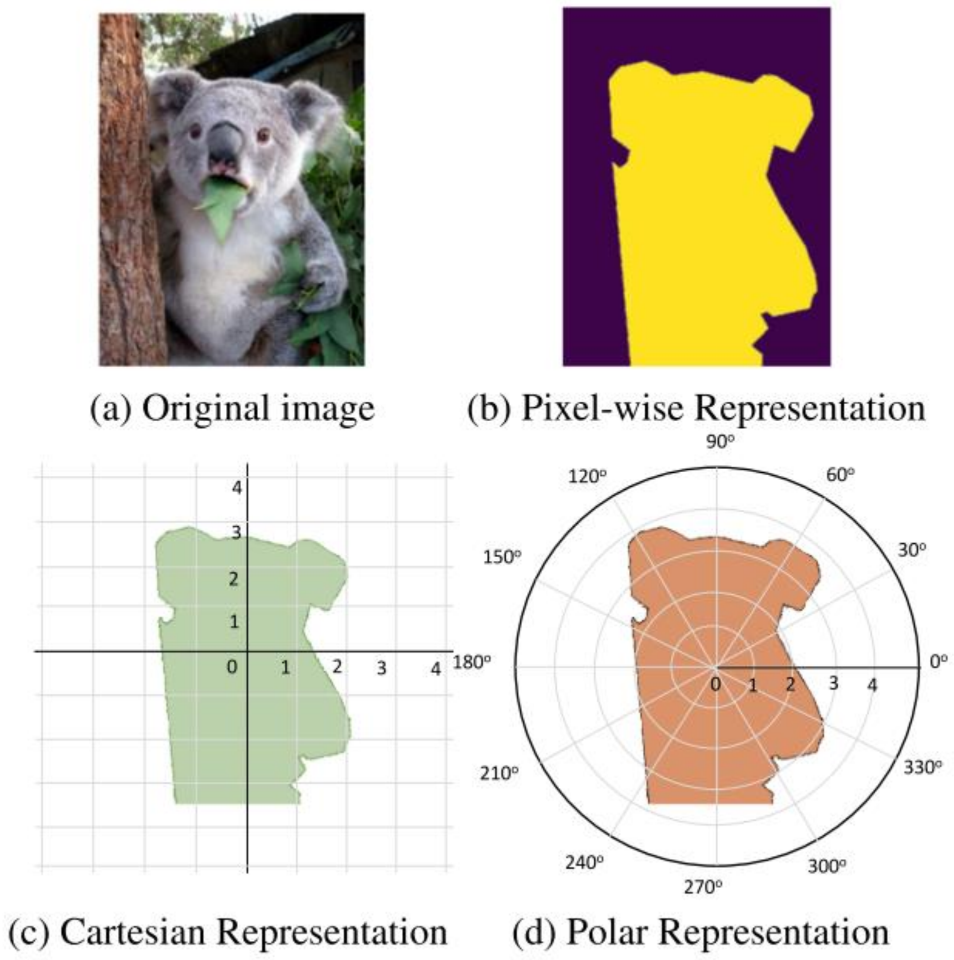
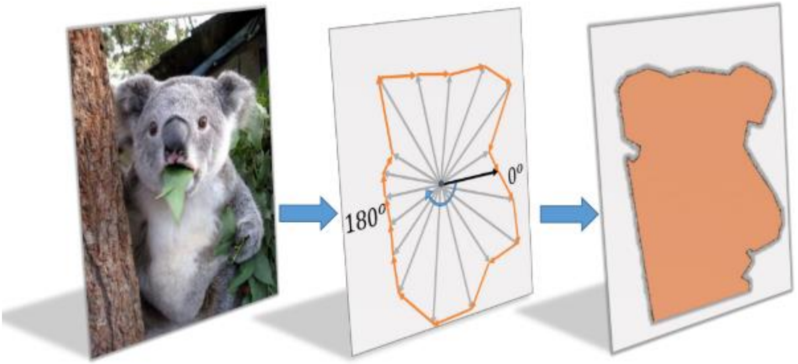


Figure 1 – Instance segmentation with different mask representations. (a) is the original image. (b) is the pixel-wise mask representation. (c) and (d) represent a mask by its contour, in the Cartesian and Polar coordinates, respectively.

边缘的极坐标表示

- **定义**：首先对任一实体，采样得到一个重心(xc, yc)和在边缘上的等夹角间距的点(xi, yi), i = 1, 2, ..., N, 其中Δθ (e.g., n = 36, Δθ = 10°)是夹角间距，计算出每一个点对(xc, yc) (xi, yi)之间的距离{d1, d2, . . . , dn}
- **重心的合理性**： Here we verify the upper bound of box center and mass-center and conclude that mass-center is more advantageous.
- **采样原则**： 距离重心1.5× strides 的范围内是正样本，否则是负样本
- 采样是为了增加样本多样性；同时重心并不总是最优选择
- **极坐标的特别情况**：
  - 如果射线有多个交点→直接取maximum
  - 如果部分角度没有交点→用10^{-6}赋值给这些d
- 作者认为：这些特别情况制约了极坐标法的性能，但是极坐标法仍然优于非参数的点对点分类
- **问题难点**：作者认为其创新不只是提出了这种极坐标，解决极坐标下的回归问题也不容易
  - **Loss平衡**：因为n个点的回归需要稠密的预测，这和分类问题相比有更大的loss
  - **相对性**：n个点不应该是孤立回归的，比方说经过一定的扰动，使点和点没有很好对应，但分割结果仍然是正确的

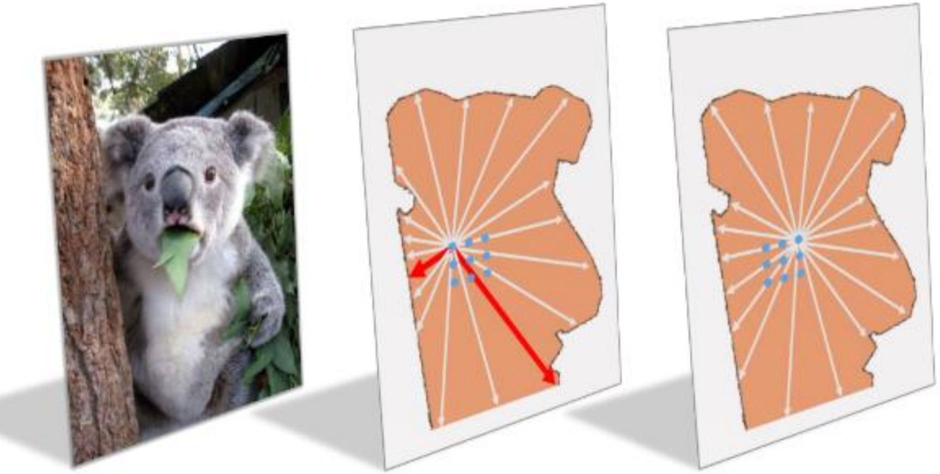
Mask 重组和极坐标Centerness



$$x_i = \cos\theta_i \times d_i + x_c$$

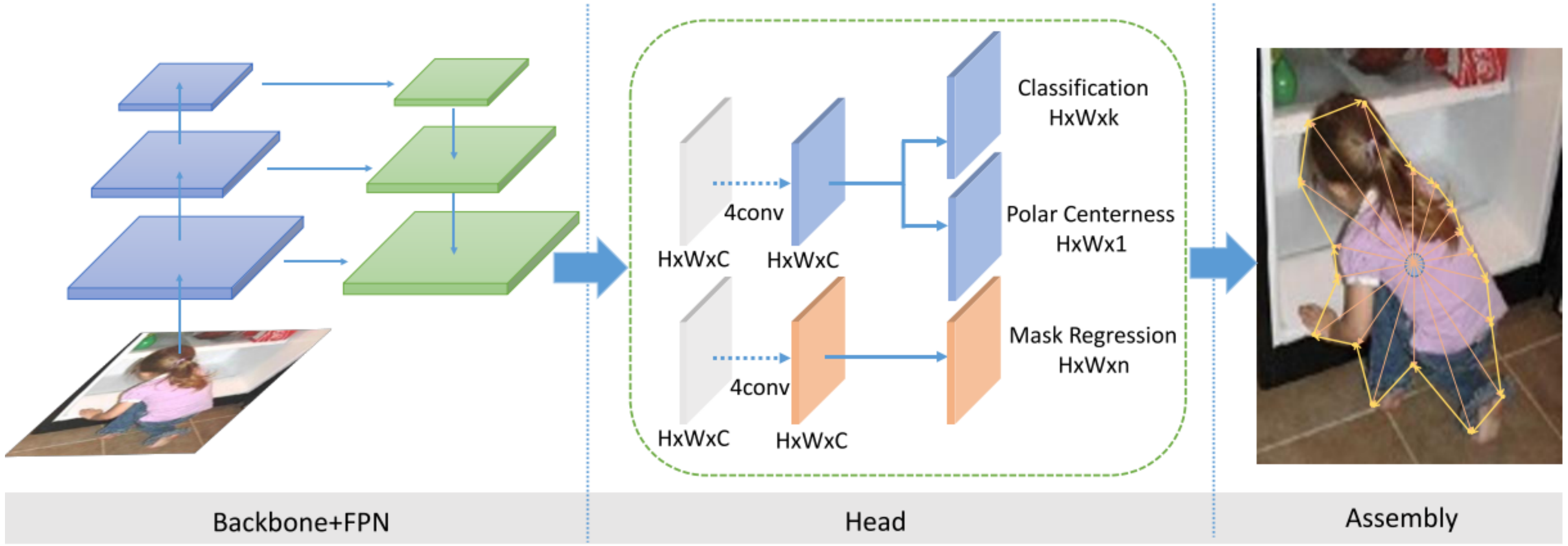
$$y_i = \sin\theta_i \times d_i + y_c.$$

$$\text{Polar Centerness} = \sqrt{\frac{\min(\{d_1, d_2, \ldots, d_n\})}{\max(\{d_1, d_2, \ldots, d_n\})}}$$



- **重组**：Given a center sample (xc, yc) and n ray's length {d1, d2, . . . , dn}，容易计算出对应的直角坐标
- **NMS:** 在重组输出的分割后，计算了输出分割的包围框，然后用NMS 进行归并
- **Centerness**: suppress these low-quality detected objects without introducing any hyper- parameters
  - the closer dmin and dmax are, higher **weight** the point is assigned
- **改进网络**：加入了一个平行的输出，用来预测**Polar Centerness of a location**
  - 输出的Centerness分数会乘到预测分数上
  - Centerness improves accuracy especially under **stricter localization metrics**, such as AP75

网络结构和Polar IOU loss



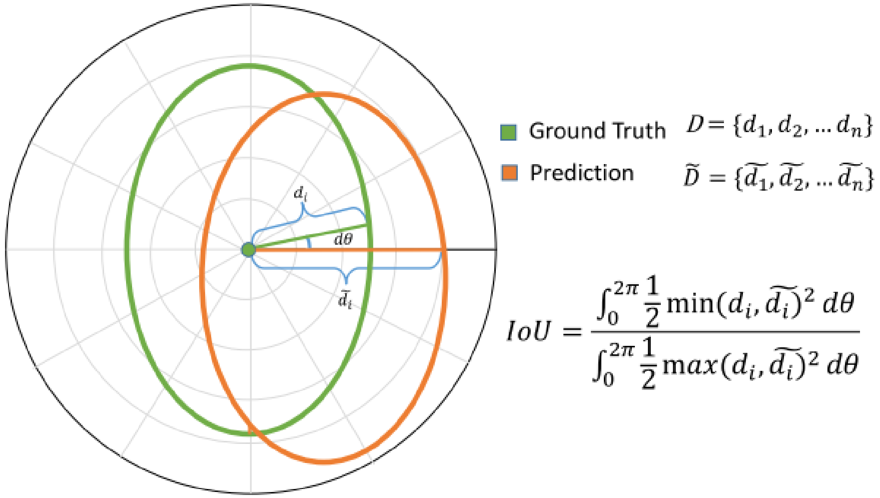


Figure 5 – Mask IoU in Polar Representation. Mask IoU (interaction area over union area) in the polar coordinate can be calculated by integrating the differential IoU area in terms of differential angles.

- **网络结构**：在基础的检测网络上，增加分类head、centerness head、**mask head**
- **损失函数**：一般的分割网络都采用l1 loss 或者iou loss
  - 定义的Polar iou loss：极坐标下用积分定义，**量化**后得到 $\mathrm{IoU} = \lim_{N\to\infty} \frac{\sum_{i=1}^{N} \frac{1}{2} d_{\min}^2 \Delta\theta_i}{\sum_{i=1}^{N} \frac{1}{2} d_{\max}^2 \Delta\theta_i}$
  - 继续量化Δθ = 2π/N则有 $\text{Polar IoU} = \frac{\sum_{i=1}^{n} d_{\min}}{\sum_{i=1}^{n} d_{\max}}$
  - 因为gt的iou是1，所以可以进行类似于cross entropy的处理 $\text{Polar IoU Loss} = \log \frac{\sum_{i=1}^{n} d_{\max}}{\sum_{i=1}^{n} d_{\min}}$

- **Advantageous properties:** differentiable, predicts the regression targets as a whole, keep the balance between classification loss and regression loss

Experiments on COCO benchmark

rays	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
18	26.2	48.7	25.4	11.8	28.2	38.0
24	27.3	49.5	26.9	12.4	29.5	40.1
36	**27.7**	49.6	27.4	12.6	30.2	39.7
72	27.6	49.7	27.2	12.9	30.0	39.7

(a) Number of Rays: More rays bring a large gain, while too many rays saturate since it already depicts the mask ground-truth well.

loss	α	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Smooth-l_1	0.05	24.7	47.1	23.7	11.3	26.7	36.8
	0.30	25.1	46.4	24.5	10.6	27.3	37.3
	1.00	20.2	37.9	19.6	8.6	20.6	31.1
Polar IoU	1.00	**27.7**	49.6	27.4	12.6	30.2	39.7

(b) Polar IoU Loss *vs*. Smooth-L1 Loss: Polar IoU Loss outperforms Smooth-l_1 loss, even the best variants of balancing regression loss and classification loss.

centerness	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Original	27.7	49.6	27.4	12.6	30.2	39.7
Polar	**29.1**	49.5	29.7	12.6	31.8	42.3

(c) Polar Centerness *vs*. Centerness: Polar Centerness bring a large gain, especially high IoU AP_{75} and large instance AP_L.

box branch	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
w	27.7	49.6	27.4	12.6	30.2	39.7
w/o	27.5	49.8	27.0	13.0	30.0	40.0

(d) Box Branch: Box branch makes no difference to performance of mask prediction.

backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-50	29.1	49.5	29.7	12.6	31.8	42.3
ResNet-101	30.4	51.1	31.2	13.5	33.5	43.9
ResNeXt-101	32.6	54.4	33.7	15.0	36.0	47.1

(e) Backbone Architecture: All models are based on FPN. Better backbones bring expected gains: deeper networks do better, and ResNeXt improves on ResNet.

scale	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	FPS
400	22.9	39.8	23.2	4.5	24.4	41.7	26.3
600	27.6	47.5	28.3	9.8	30.1	43.1	21.7
800	29.1	49.5	29.7	12.6	31.8	42.3	17.2

(f) Accuracy/speed trade-off on ResNet-50: PolarMask performance with different image scales. The FPS is reported on one V100 GPU.

结果展示

Figure 6 – Visualization of PolarMask with Smooth-l_1 loss and Polar IoU loss. Polar IoU Loss achieves to regress more accurate contour of instance while Smooth-l_1 Loss exhibits systematic artifacts.

讨论

- 射线越多，应该性能越好才对?
- 大物体的边缘容易描述不清，如何解决?
- 3D条件下拓展是否合适?

PointFlow: 3D Point Cloud Generation with Continuous Normalizing Flows（ICCV2019）

Guandao Yang[1,2*], Xun Huang[1,2*], Zekun Hao[1,2], Ming-Yu Liu[3], Serge Belongie[1,2], Bharath Hariharan[1]
[1]Cornell University [2]Cornell Tech [3]NVIDIA

任务：三维点云生成，**凭空产生**符合要求的点云，https: //github.com/stevenygd/PointFlow.

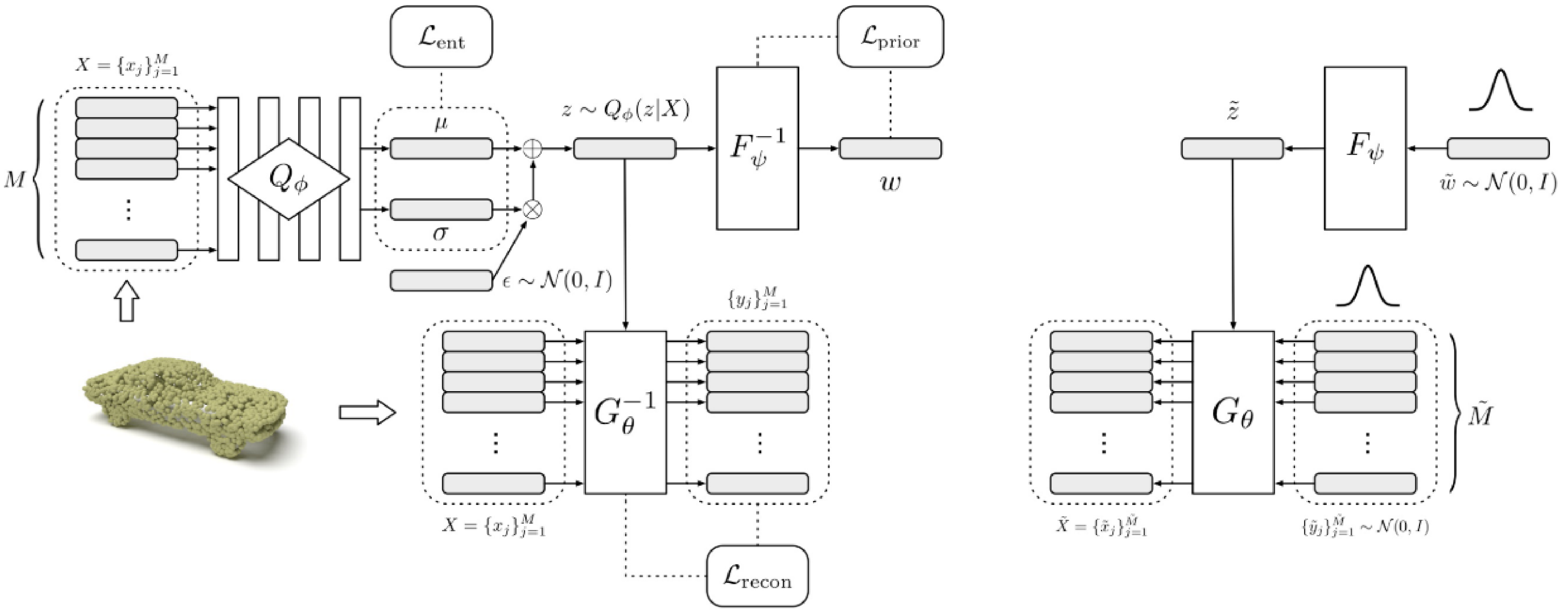
应用：拓充点云数据集，autoencoder编码已有的数据，但本文没有尝试从一张图片生成点云的工作

PointFlow：一个在分布上生成分布的的两级生成器；第一级生成一个形状，第二级生成形状的每一个点

相关工作：

- 点云深度学习：目前分类、分割、关键点提取任务都已有方案，但多数都规定了点云结构是N*3。N不可变是一个问题，此外目前的方案没有考虑点云微扰下的形状不变形，引入不必要的损失值
- 生成模型：主流方案有GAN、VAE、**auto-regressive models**、**flow-based models；**后两者能够计算似然函数，且**flow-based models**计算代价小，在图像、视频生成中已经有应用

PointFlow方案



(a) Training (Auto-encoding)　　　　(a) Test (Sampling)

- Our generative model should be able to both **sample shapes** and **sample an arbitrary number of points** from a shape.
- 训练：训练编码器Q，和解码器G，F
- 测试：采样w和y，获得z和x

Continuous normalizing flow

- 如果$f1, ..., fn$是一系列可逆的变换，并且有$x = fn \circ fn-1 \circ \cdots \circ f1(y)$，x是y经过这些变换得到的
- 概率描述则有这样的关系：$\log P(x) = \log P(y) - \sum_{k=1}^{n} \log|\det \frac{\delta f_k}{\delta y_{k-1}}|$，这里的$|\det \frac{\delta f_k}{\delta y_{k-1}}|$部分可以用深度网络建模，方便地输出
- 根据CNF模型，$x = y(t0) + \int_{t0}^{t1} f(y(t), t)dt, y(t0) \sim P(y)$
- 可以推出$\log P(x) = \log P(y(t0)) - \int_{t0}^{t1} Tr\left(\frac{\delta f}{\delta y(t)}\right)dt$
- 这部分可以采用ordinary differential equation (ODE)工具计算得到结果（NIPS2018）

Variational auto-encoder

- 我们的目标是最大化$\log P\theta(X)$，但直接对其处理比较困难，所以我们最大化它的下界——evidence lower bound (ELBO)

$$\begin{aligned}\log P_\theta(X) &\geq \log P_\theta(X) - D_{KL}(Q_\phi(z|X)||P_\theta(z|X)) \\ &= \mathbb{E}_{Q_\phi(z|x)}\left[\log P_\theta(X|z)\right] - D_{KL}\left(Q_\phi(z|X)||P_\psi(z)\right) \\ &\triangleq \mathcal{L}(X;\phi,\psi,\theta)\,, \qquad (3)\end{aligned}$$

- 假设我们z是x的一个先验，那么原式的ELBO就等价于z条件下出现这个X的概率，减去X映射出z的分割和z的先验分布的KL距离
- 第一项可以用采样z的方式得到概率值，第二项可以用z的参数和先验计算kl距离
- Q是编码器，输入X得到对应的先验z的分布参数（均值、方差）

Flow-based **point generation** from shape representations

- 之前已经推出$\log P(x) = \log P(y(t0)) - \int_{t0}^{t1} Tr\left(\frac{\delta f}{\delta y(t)}\right) dt$
- 设G是一个形变器，它从初始点y映射到x，这里y(t1) = x

$$x = G_\theta(y(t_0); z) \triangleq y(t_0) + \int_{t_0}^{t_1} g_\theta(y(t), t, z) dt, y(t_0) \sim P(y),$$

- 因此可以有 $$\log P_\theta(x|z) = \log P(G_\theta^{-1}(x; z)) - \int_{t_0}^{t_1} \mathrm{Tr}\left(\frac{\partial g_\theta}{\partial y(t)}\right) dt. \tag{5}$$

$$\begin{aligned}\log P_\theta(X) &\geq \log P_\theta(X) - D_{KL}(Q_\phi(z|X)||P_\theta(z|X)) \\ &= \boxed{\mathbb{E}_{Q_\phi(z|x)}[\log P_\theta(X|z)]} - D_{KL}(Q_\phi(z|X)||P_\psi(z)) \\ &\triangleq \mathcal{L}(X; \phi, \psi, \theta),\end{aligned} \tag{3}$$

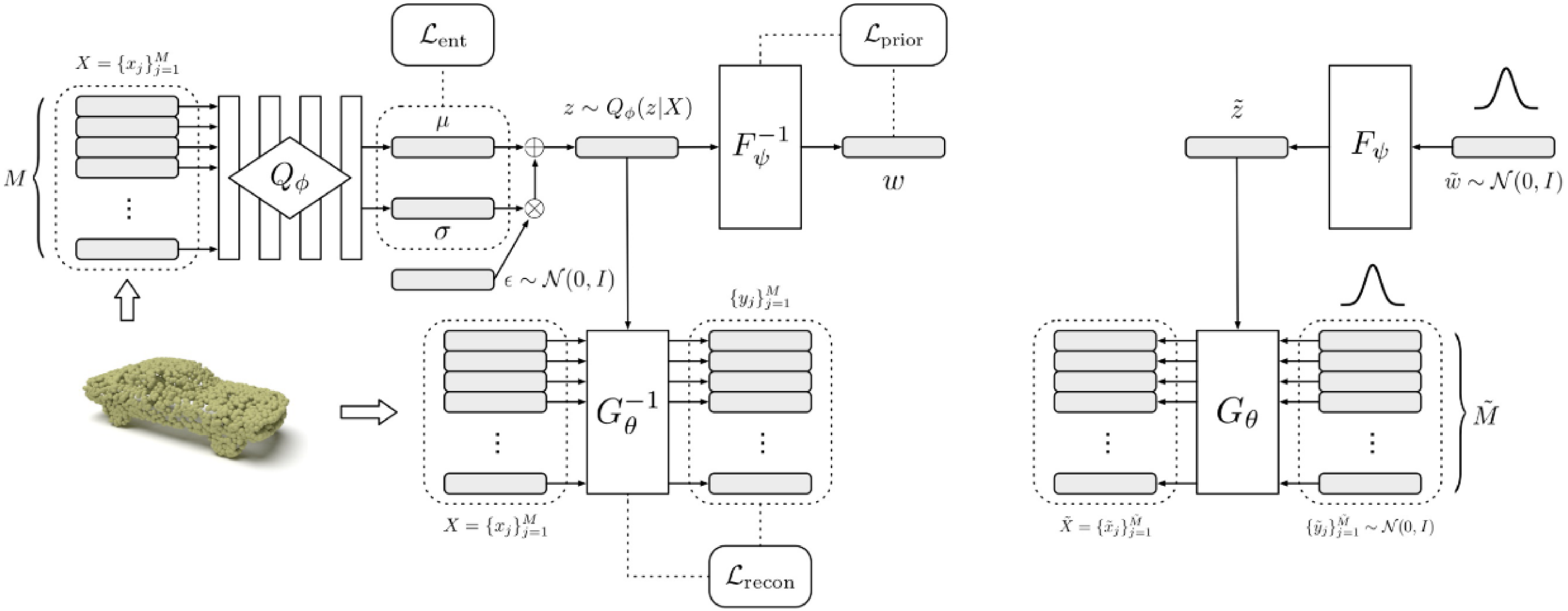
Flow-based prior over shapes

$$\begin{aligned}\log P_\theta(X) &\geq \log P_\theta(X) - D_{KL}(Q_\phi(z|X)||P_\theta(z|X)) \\ &= \mathbb{E}_{Q_\phi(z|x)}[\log P_\theta(X|z)] - \boxed{D_{KL}(Q_\phi(z|X)||P_\psi(z))} \\ &\triangleq \mathcal{L}(X;\phi,\psi,\theta)\,, \end{aligned} \quad (3)$$

- $D_{KL}(Q\varphi(z|x)||P\psi(z)) = -E_{Q\varphi(z|x)}[\log P\psi(z)] - H[Q\varphi(z|X)]$
- 这里H是熵， $P\psi(z)$是标准正态先验，继续用CNF定义z

$$z = F_\psi(w(t_0)) \triangleq w(t_0) + \int_{t_0}^{t_1} f_\psi(w(t),t)dt,\, w(t_0) \sim P(w)\,,$$

- 也可以得到 $\log P_\psi(z) = \log P\left(F_\psi^{-1}(z)\right) - \int_{t_0}^{t_1} \mathrm{Tr}\left(\frac{\partial f_\psi}{\partial w(t)}\right) dt\,.$
- 从而解决这里的第一项$-E_{Q\varphi(z|x)}[\log P\psi(z)]$



(a) Training (Auto-encoding)

(a) Test (Sampling)

$$
\begin{aligned}
\mathcal{L}(X;\phi,\psi,\theta) &= \mathbb{E}_{Q_\phi(z|x)}\left[\log P_\psi(z) + \log P_\theta(X|z)\right] + H[Q_\phi(z|X)] \\
&= \mathbb{E}_{Q_\phi(z|X)}[\log P\left(F_\psi^{-1}(z)\right) - \int_{t_0}^{t_1} \operatorname{Tr}\left(\frac{\partial f_\psi}{\partial w(t)}\right) dt \\
&+ \sum_{x\in X} (\log P(G_\theta^{-1}(x;z)) - \int_{t_0}^{t_1} \operatorname{Tr}\left(\frac{\partial g_\theta}{\partial y(t)}\right) dt)] \\
&+ H[Q_\phi(z|X)]\,. \qquad (8)
\end{aligned}
$$

Experiments

Table 1: Generation results. ↑: the higher the better, ↓: the lower the better. The best scores are highlighted in bold. Scores of the real shapes that are worse than some of the generated shapes are marked in gray. MMD-CD scores are multiplied by 10^3; MMD-EMD scores are multiplied by 10^2; JSDs are multiplied by 10^2.

		# Parameters (M)		JSD (↓)	MMD (↓)		COV (%, ↑)		1-NNA (%, ↓)	
Category	Model	Full	Gen		CD	EMD	CD	EMD	CD	EMD
Airplane	r-GAN	7.22	6.91	7.44	0.261	5.47	42.72	18.02	93.58	99.51
	l-GAN (CD)	1.97	1.71	4.62	0.239	4.27	43.21	21.23	86.30	97.28
	l-GAN (EMD)	1.97	1.71	**3.61**	0.269	3.29	**47.90**	**50.62**	87.65	85.68
	PC-GAN	9.14	1.52	4.63	0.287	3.57	36.46	40.94	94.35	92.32
	PointFlow (ours)	**1.61**	**1.06**	4.92	**0.217**	**3.24**	46.91	48.40	**75.68**	**75.06**
	Training set	-	-	6.61	0.226	3.08	42.72	49.14	70.62	67.53
Chair	r-GAN	7.22	6.91	11.5	2.57	12.8	33.99	9.97	71.75	99.47
	l-GAN (CD)	1.97	1.71	4.59	2.46	8.91	41.39	25.68	64.43	85.27
	l-GAN (EMD)	1.97	1.71	2.27	2.61	**7.85**	40.79	41.69	64.73	65.56
	PC-GAN	9.14	1.52	3.90	2.75	8.20	36.50	38.98	76.03	78.37
	PointFlow (ours)	**1.61**	**1.06**	**1.74**	**2.42**	7.87	**46.83**	**46.98**	**60.88**	**59.89**
	Training set	-	-	1.50	1.92	7.38	57.25	55.44	59.67	58.46
Car	r-GAN	7.22	6.91	12.8	1.27	8.74	15.06	9.38	97.87	99.86
	l-GAN (CD)	1.97	1.71	4.43	1.55	6.25	38.64	18.47	63.07	88.07
	l-GAN (EMD)	1.97	1.71	2.21	1.48	5.43	39.20	39.77	69.74	68.32
	PC-GAN	9.14	1.52	5.85	1.12	5.83	23.56	30.29	92.19	90.87
	PointFlow (ours)	**1.61**	**1.06**	**0.87**	**0.91**	**5.22**	**44.03**	**46.59**	**60.65**	**62.36**
	Training set	-	-	0.86	1.03	5.33	48.30	51.42	57.39	53.27

CD：参考和输出分别计算最近匹配点距离，然后加合

EMD：只计算输出点双射距离

Jensen-Shannon Divergence (JSD)：将参考集和输出集平均值量化后计算KL距离

$$\mathrm{COV}(S_g, S_r) = \frac{|\{\arg\min_{Y \in S_r} D(X,Y) | X \in S_g\}|}{|S_r|}$$

$$\mathrm{MMD}(S_g, S_r) = \frac{1}{|S_r|} \sum_{Y \in S_r} \min_{X \in S_g} D(X,Y)$$

$$\text{1-NNA}(S_g, S_r) = \frac{\sum_{X \in S_g} \mathbb{I}[N_X \in S_g] + \sum_{Y \in S_r} \mathbb{I}[N_Y \in S_r]}{|S_g| + |S_r|}$$

讨论

- 是否可以用此方案进行点云生成?
- 是否可以将先验改成一副图像