A Generalist Agent——hoho

研究方法

本文创建了一个通用的模型——Gato，致力于打造一个通用的人工智能模型

数据处理

- 数据来源是多模的，包含图片、文字、体感数据（proprioception）、机械臂动作数据（joint toques）、游戏数据（button presses），等等离散和连续的环境观测数据和动作数据。数据组成如下图：

Table 1 | **Datasets.** Left: Control datasets used to train Gato. Right: Vision & language datasets. Sample weight means the proportion of each dataset, on average, in the training sequence batches.

Control environment	Tasks	Episodes	Approx. Tokens	Sample Weight
DM Lab	254	16.4M	194B	9.35%
ALE Atari	51	63.4K	1.26B	9.5%
ALE Atari Extended	28	28.4K	565M	10.0%
Sokoban	1	27.2K	298M	1.33%
BabyAI	46	4.61M	22.8B	9.06%
DM Control Suite	30	395K	22.5B	4.62%
DM Control Suite Pixels	28	485K	35.5B	7.07%
DM Control Suite Random Small	26	10.6M	313B	3.04%
DM Control Suite Random Large	26	26.1M	791B	3.04%
Meta-World	45	94.6K	3.39B	8.96%
Procgen Benchmark	16	1.6M	4.46B	5.34%
RGB Stacking simulator	1	387K	24.4B	1.33%
RGB Stacking real robot	1	15.7K	980M	1.33%
Modular RL	38	843K	69.6B	8.23%
DM Manipulation Playground	4	286K	6.58B	1.68%
Playroom	1	829K	118B	1.33%
Total	596	63M	1.5T	85.3%

Vision / language dataset	Sample Weight
MassiveText	6.7%
M3W	4%
ALIGN	0.67%
MS-COCO Captions	0.67%
Conceptual Captions	0.67%
LTIP	0.67%
OKVQA	0.67%
VQAV2	0.67%
Total	14.7%

- 数据编码（Tokenization和Embedding）

1. 文字数据通过SentencePiece方法切分为32000个子词；
2. 图片转换为一个个无区域覆盖的16x16像素区域（patch）的序列，按照raster order（按行优先排序），每个patch的像素正则化为[-1, 1]区间的数字，并除以patch的大小的开方（$\sqrt{16} = 4$）；
3. 离散数据，譬如button presses，“平铺“（flatten）为一行的整型序列，映射到[0, 1024]范围；

4. 连续数据，譬如proprioception、joint torque，先flatten为一行浮点数序列，使用mu-law方法编码到[-1, 1]范围：$F(x) = sgn(x)\frac{log(|x|\mu+1.0)}{log(M\mu+1.0)}, \mu = 100, M = 256$，然后将这些数据均匀离散到1024个方格内，将它们映射为整型数并偏移到[32000, 33024]内；
5.

然后，文字、离散和连续的token会embed为一个向量，并加上它们的position embedding。而图像token会通过ResNet提取为特征向量，也加上其patch position encoding.

postion编码方法如下:

图像的position编码

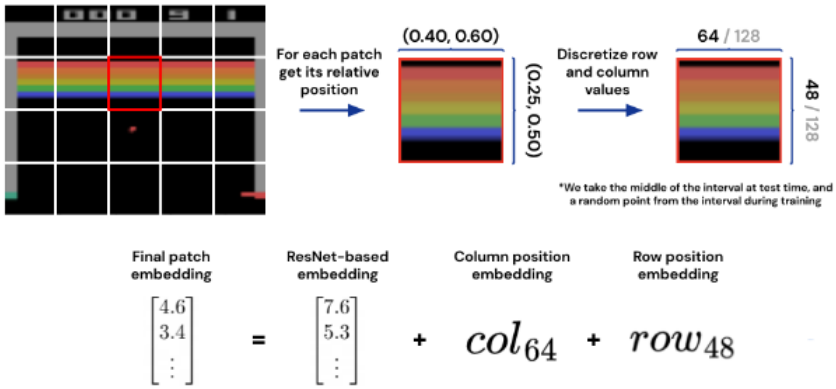
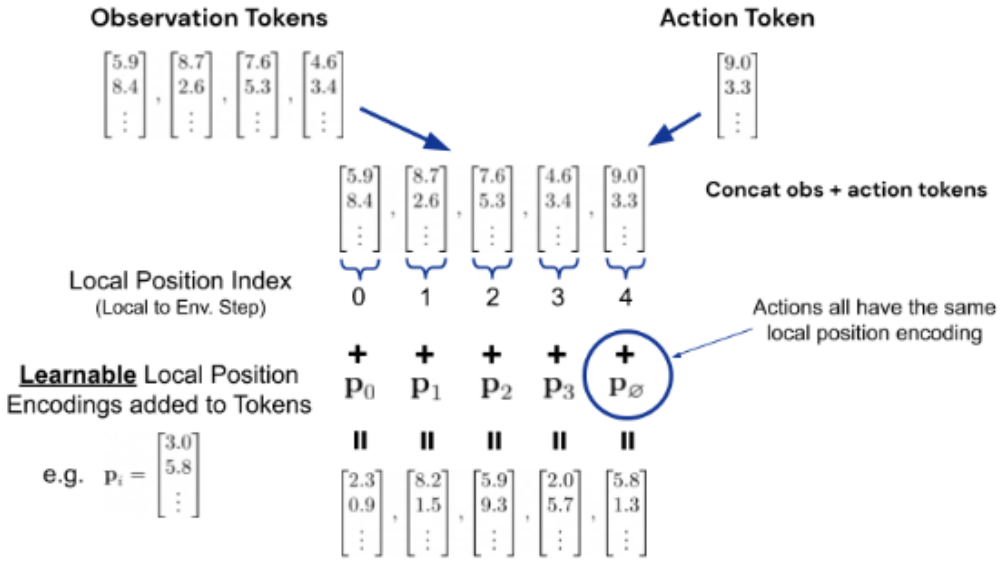


Figure 15 | **Patch position encodings.** Calculating patch position encodings (red) within the global image (far left). The relative row and column positions (i.e. positions normalized between [0, 1]) are first discretized using uniform binning and used to index a learnable row and column position encoding. These two encodings are then added to the token embedding corresponding to the patch.

其他的position编码



最终数据样例如下图：

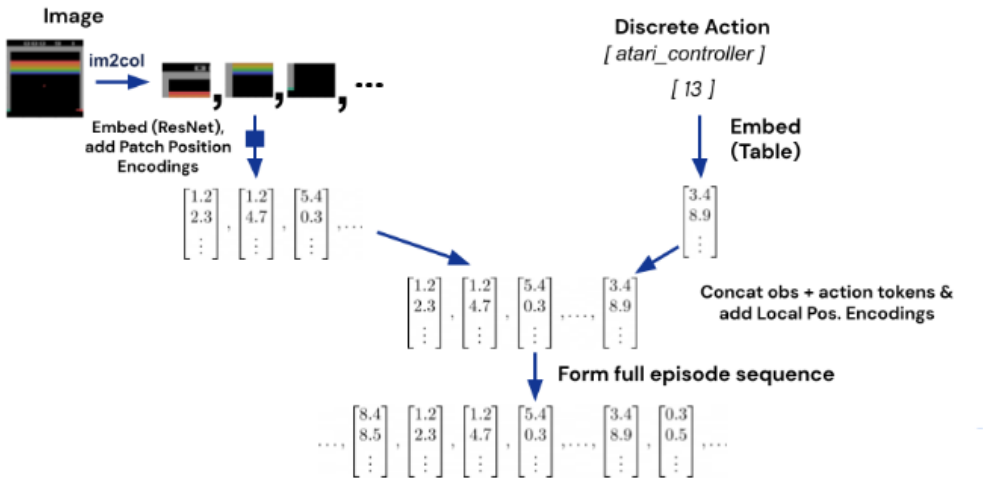


Figure 12 | **A visualization of tokenizing and sequencing images and discrete values.**

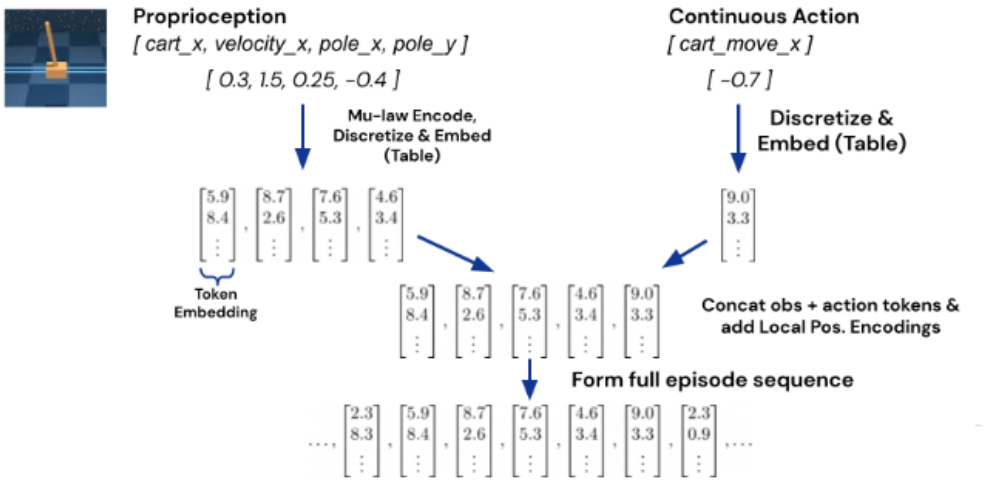


Figure 13 | A visualization of tokenizing and sequencing continuous values, e.g. proprioception.

这样，每个时间步的数据形如$[y_{1:k}, x_{1:m}, z_{1:n}, '|', a_{1:A}]$，其中：

- $y_{1:k}$为文字embeding
- $x_{1:m}$为图像encoding
- $z_{1:n}$为离散或连续向量的encoding
- '|'为分隔符，分隔观测与动作数据
- $a_{1:A}$为动作向量

每个回合（episode）的数据形如：

$$s_{1:L} = \{[y^1_{1:k}, x^1_{1:m}, z^1_{1:n}, '|', a^1_{1:A}], ..., [y^T_{1:k}, x^T_{1:m}, z^T_{1:n}, '|', a^T_{1:A}]\}$$

对于文字、离散或连续与动作组合的数据，可以以自回归方式（下一个数据作为上一个数据的label）让模型学习，对于图像和动作组合的数据，暂无作为模型预测结果（论文指出有待研究）。

网络模型

根据链式法：$logp_\theta(s1, s2, ..., s_L) = \sum_l^L logp_\theta(sl|s1, ..., s_{l-1})$

定义函数m(b,t)，表示在index t的token如果是文字或者是经过log后的动作，则为1，否则为0。

损失函数定义为：

$$L(\theta, B) = -\sum_{b=1}^{|B|}\sum_{l=1}^{L} m(b,t) log p_\theta(s_l^{(b)} | s_1^{(b)}, ..., s_{l-1}^{(b)})$$

网络主要使用transformer的decoder部分。

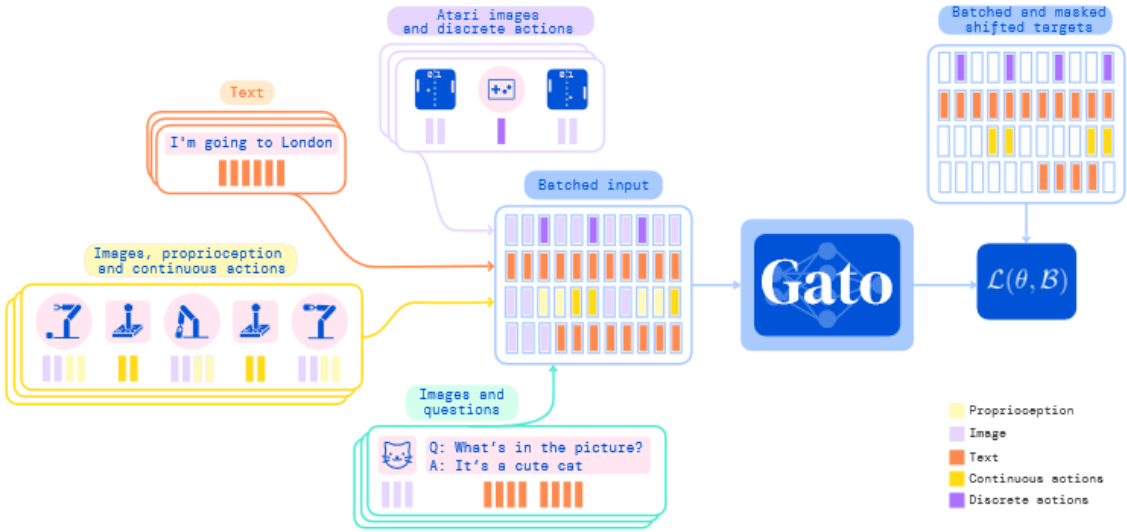
结构如下：

Table 5 | **Gato transformer hyperparameters.**

Hyperparameter	Gato 1.18B	364M	79M
Transformer blocks	24	12	8
Attention heads	16	12	24
Layer width	2048	1536	768
Feedforward hidden size	8192	6144	3072
Key/value size	128	128	32
Shared embedding	True		
Layer normalization	Pre-norm		
Activation Function	GEGLU		

网络输出下一个离散token的分布。

整体架构如下：

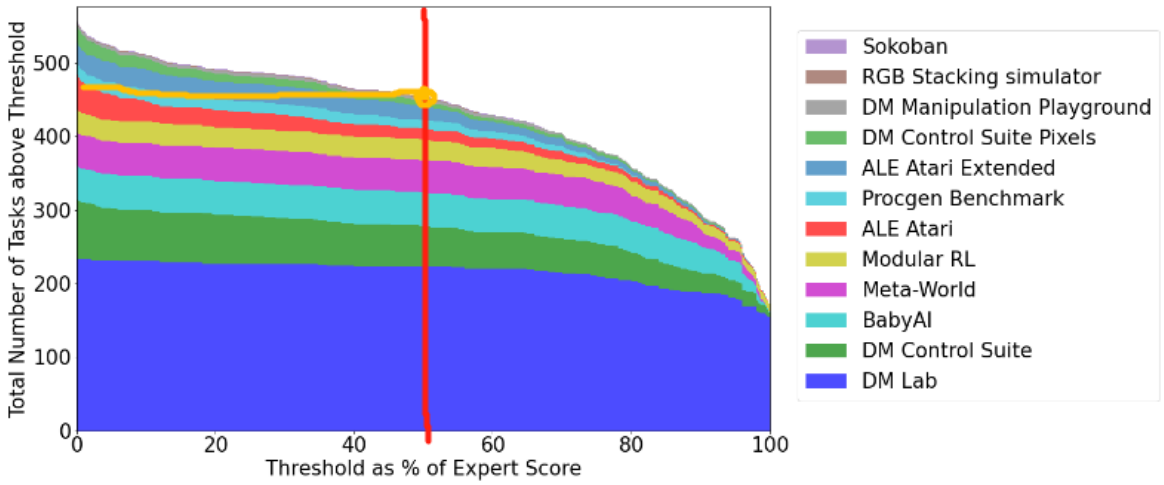


另外，为了消除一些任务的歧义，模型需要更深层的上下文信息。论文运用prompt conditioning技术。

训练时，对于每个batch的25%序列，预先设定一个prompt序列，这个序列都来自同一个任务的同一个agent产生的一个回合。其中一半的prompt序列来自回合的结尾，作为是各种领域的目标条件的一种形式（acting as a form of goal conditioning for many domains.），另外一半则均匀的从回合中抽样。

验证时，智能体就可以被prompt通过使用一个预期任务的成功证明（the agent can be prompted using a successful demonstration of the desired task.）

研究成果



上图表示Gato进行任务的水平等于或超过专家分数的任务数量。X轴表示专家进行某个任务的得分，Y轴每个任务的颜色带的宽度表示任务的某个数量。可见Gato在604个任务中有大约450个任务的得分是超过50%的专家得分。

研究结论

（暂无）

附

疑问

1. 奖励函数呢？如何体现回报

启示

1. transformer在强化学习中的应用