Probabilistic Attention for Interactive Segmentation: Supplementary Material

Prasad Gabbur
Apple
pgabbur@apple.com

Manjot Bilkhu
Apple
mbilkhu@apple.com

Javier Movellan
Apple
movellan@apple.com

A Graphical model

The constrained generative model equivalent to standard dot-product attention (Section 3.3) can be expressed as a Bayesian probabilistic graphical model shown in Fig 1. In order to generate a query q_i (observed) and a value v_i (observed) at unit i, a memory unit u_i (unobserved) is first sampled from a prior π_{ij} over units j in the the memory bank. This is done independently for each unit across the memory bank comprising of n total units. The per-unit queries and values are then sampled independently from isotropic Gaussians as described in Section 3.3.

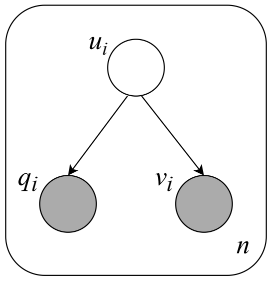


Figure 1: **Probabilistic generative model for queries and values**. Graphical representation of the generative model for a query (q_i) and a value (v_i) through a corresponding hidden latent variable (u_i) that indexes over units of a probabilistic memory bank. n denotes the #units in the memory bank as well as the number of generated query/value pairs.

B Proofs

B.1 Relationship to standard attention

We provide a detailed proof of Eq. (12). With the assumptions of Section 3.3, Eq. (4) is given by

$$Q_i(v^t, v^{t+1}) = \sum_j w_{i,j}^t \left(\frac{d}{2} \log\left(\frac{\alpha_j}{2\pi}\right) - \frac{\alpha_j}{2}\|q - \xi_j\|^2 + \frac{m}{2}\log\left(\frac{\beta_j}{2\pi}\right) - \frac{\beta_j}{2}\|v^{t+1} - \mu_j\|^2 \right), \tag{B.1}$$

where $w_{i,j}^t$, including the precision parameter β_j, is

$$w_{i,j}^t = \frac{\pi_{i,j}\, \beta_j\, e^{-\frac{\alpha_j}{2}\|q-\xi_j\|^2}\, e^{-\frac{\beta_j}{2}\|v^t-\mu_j\|^2}}{\sum_j \pi_{i,j}\beta_j\, e^{-\frac{\alpha_j}{2}\|q-\xi_j\|^2}\, e^{-\frac{\beta_j}{2}\|v^t-\mu_j\|^2}}. \tag{B.2}$$

Taking the derivative w.r.t. v^{t+1} and setting it to zero,

$$\nabla_{v^{t+1}} Q_i(v^t, v^{t+1}) = \sum_j w_{i,j}^t \beta_j (\mu_j - v^{t+1}) = 0, \tag{B.3}$$

the EM update equation reduces to

$$v^{t+1} = \sum_j w_{i,j}^t \, \mu_j. \tag{B.4}$$

The prior of Eq. (10) makes $w_{i,j}^t$ independent of i (permutation equivariant) and simplifies the optimal value inference equation to

$$v^{t+1} = \sum_j w_{i,j}^t \mu_j \tag{B.5}$$

$$w_{i,j}^t = \frac{e^{\alpha \xi_j^T q} \; e^{\beta \mu_j^T v_t}}{\sum_j e^{\alpha \xi_j^T q} \;\; e^{\beta \mu_j^T v_t}}. \tag{B.6}$$

It is easy to see that as $\beta \to 0$ we obtain the standard dot product attention update as in Eq. (12).

B.2 Online key adaptation

At any EM iteration t, the auxiliary function $Q(\xi_{1:n}^t, \xi_{1:n}^{t+1})$ for key update is given by

$$Q(\xi_{1:n}^t, \xi_{1:n}^{t+1}) = \log p(\xi_{1:n}^{t+1}) + \sum_{i=1}^n \sum_{j=1}^n w_{i,j}^t \log p_j(q_i, u_j \mid \xi_j^{t+1}), \tag{B.7}$$

where

$$w_{i,j}^t = p_i(u_j \mid q_i, \xi_{1:n}^t) = \frac{\pi_{i,j} \, p(q_i \mid u_j, \xi_j^t)}{\sum_{k=1}^n \pi_{i,k} \, p(q_i \mid u_k, \xi_k^t)}. \tag{B.8}$$

Taking the derivative w.r.t. the key vector ξ_k^{t+1} of unit k

$$\nabla_{\xi_k^{t+1}} Q(\xi_{1:n}^t, \xi_{1:n}^{t+1}) = \theta_\xi (\xi_k^t - \xi_k^{t+1}) + \sum_{i=1}^s w_{i,k}^t \alpha_k (q_i - \xi_k^{t+1}). \tag{B.9}$$

Setting the derivative to zero and solving for ξ_k^{t+1} leads to the online key adaptation update of Eq.(18).

B.3 Online adaptation of α_j precision parameters

The precision parameters α_j in the per-unit query likelihoods can be adapted online based on the observed queries, similar to keys in Eq. (18). In order to avoid overfitting to the observed queries, we use a Gamma prior with parameters $\theta_{\alpha,1}, \theta_{\alpha,2}$, which leads to the following update for the precisions

$$\alpha_k^{t+1} = \frac{\theta_{\alpha,1} + d/2 \sum_{i=1}^n w_{i,k}^t - 1}{\theta_{\alpha,2} + \sum_{i=1}^n w_{i,k}^t \, \frac{1}{2} \|q_i - \xi_k\|^2}. \tag{B.10}$$

B.4 Updates of value likelihood parameters based on fixed values

In addition to value propagation (Eq. (23)), the probabilistic attention model allows updating the per-unit value likelihood component parameters based on the information provided by the fixed pre-selected values. Specifically, the EM updates for the unit k likelihood parameters β_k and $\pi_{i,k}$ are given by

$$\beta_k^{t+1} = \frac{\theta_{\beta,1} + d/2 \sum_{i=1}^s w_{i,k}^t - 1}{\theta_{\beta,2} + \frac{1}{2} \sum_{i=1}^s w_{i,k}^t \|v_i - \mu_k\|^2} \tag{B.11}$$

$$\pi_{i,k}^{t+1} = \frac{w_{i,k}^t + \theta_{\pi,i,k} - 1}{\sum_k w_{i,k}^t + \theta_{\pi,i,k} - 1}, \tag{B.12}$$

where $\theta_{\beta,1}, \theta_{\beta,2}$ are the parameters for a Gamma prior distribution over β_k, and $\theta_{\pi,i,k}$ are Dirichlet prior parameters over $\pi_{i,k}$. The weights $w_{i,k}^t$ above are the same as in Eq. (24).

B.5 Position embedding formulations

Here we provide details on how we arrive at the form of the per-unit query likelihood of Eq. (25). By choosing to include the position embeddings r^q_{j-i} through an extra normal likelihood function, the per-unit query likelihood is given by

$$p_i(q \mid \xi_j, r^q_{j-i}, u_j) \propto \mathcal{N}(q \mid \xi_j,\, \frac{1}{\alpha_j} I_d)\mathcal{N}(q \mid r^q_{j-i},\, \frac{1}{\alpha_j} I_d),$$

where $\mathcal{N}(a \mid b, c)$ is the Gaussian likelihood function over a with mean b and covariance matrix c. I_d is a $d \times d$ identity matrix. Making use of the fact that the product of two normal likelihood functions is also a normal and completing the square

$$\mathcal{N}(q \mid \xi_j,\, \frac{1}{\alpha_j} I_d)\mathcal{N}(q \mid r^q_{j-i},\, \frac{1}{\alpha_j} I_d) = \mathcal{N}(q \mid \frac{\xi_j + r^q_{j-i}}{2},\, \frac{1}{2\alpha_j} I_d)\mathcal{N}(\xi_j \mid r^q_{j-i},\, \frac{2}{\alpha_j} I_d), \tag{B.13}$$

we arrive at the form in Eq. (25). Note that there is effectively no direct interaction between ξ_j and r^q_{j-i} terms in the above. The choice of this form of position embedding is to make our formulation equivalent to how it is encoded in contemporary works [13, 10] under the assumptions of Section 3.3. There may be other ways to encode position embeddings within our framework such as directly influencing the prior based on some distance measure $d(i, j)$ between the locations of units i and j, as given by

$$\pi_{i,j} \propto \exp\left(-d(i, j)\right). \tag{B.14}$$

C Models, training and evaluation

Details of the interactive segmentation models used in the experiments and their training and evaluation procedures are provided below.

C.1 Interactive segmentation model

We use a single model to both predict an initial mask and correct it subsequently given an input image and annotator corrections. It takes a 3 channel input RGB image and 3 additional channels, one each to encode the object bounding box, positive and negative annotator corrections respectively. The object bounding box is specified using 2 clicks to roughly correspond to the box corners (top-left+bottom-right or bottom-left+top-right). These along with the positive and negative corrective clicks provided by the annotator are encoded as binary disks of radius 8 pixels following the findings in previous works [1, 8, 7]. We experiment using different architectures: HRNetV2+OCR [14] and DeepLabV3+ [2], backbones: ResNet-50 and ResNet-101 [5], and training datasets: SBD [4] and LVIS [3]) for specific experiments.

C.2 Training

All of our models are trained following a curriculum over three tasks. The first task is to predict a mask given an input image and object bounding box but empty corrective channels. The second task is to predict a mask given the image, bounding box and the corrective channels populated with randomly sampled clicks on the object foreground (positive) and background (negative). The third task is the corrective task, which is similar to the second task but with corrective channels containing corrective clicks randomly sampled from the false positive and negative error regions of the model's prediction. For both the second and third tasks, we randomly sample 1-3 clicks and 0-3 clicks for the positive and negative channels respectively. All our models are trained using RAdam optimizer [6] on a polynomial decay learning rate schedule (power=0.9) with a base learning rate of 10^{-4} decaying to 10^{-5} in 70 epochs and constant thereafter for a total of 150 epochs. The first and second tasks of our curriculum use 20 epochs each in a sequence and the remaining epochs are used for the third task. We use a batch size of 16 and train our models over 4 NVidia Volta 2-GPU nodes with 32GB of GPU memory each using Distributed Data Parallel framework in PyTorch.

C.3 Evaluation

The trained models are evaluated on the GrabCut [11] and Berkeley [9] datasets. Specifically, we plot the improvement in mask accuracy, i.e. mean IOU relative to ground truth, as a function of the number of clicks [1], starting with the initial 2 clicks to specify a bounding box. For all the experiments, we simulate annotators by sampling the next corrective click from the largest error areas (positive and negative) obtained by comparing the ground truth mask with the model's current prediction. Note that for both training and evaluation, we crop the object bounding boxes with a finite padding around them without resizing the input image, as network input. We also add noise to the bounding box corners which introduces some variance into the evaluation metrics. In order to account for the variances in the crops, we repeat simulations over multiple trials (5 or 10) and report both the mean and the standard error across trials. We would like to point out that a better performing model reaches a certain mIoU faster, i.e. with fewer clicks.

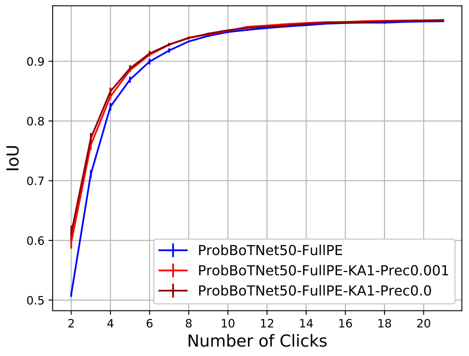
C.4 Other hyperparameters

Apart from the training hyperparameters described above, we set the value of the query/key Gaussian precisions α_j to a constant value equal to $\frac{1}{\sqrt{d}}$, where d is the query/key embedding dimension. This is similar to standard scaled dot-product attention. We use non-zero precisions β_j only to train models with value propagation (Eq. (23)). In that case they are all set equal to 0.1.

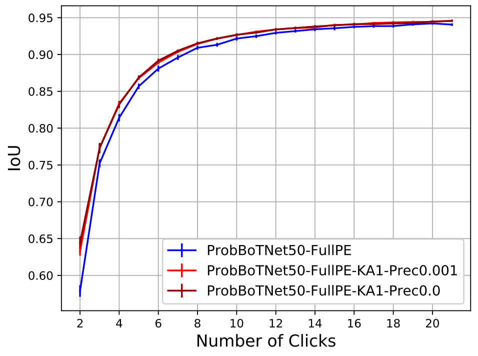
Also, key adaptation and value propagation iterations use additional hyperparameters, specifically the priors θ_ξ and θ_μ. We set θ_ξ to 0 (ML update) unless otherwise specified. In order to choose an optimum value for θ_μ, we do a grid search over a set of 4 values: [0.1, 1, 10, 100] and choose the one that results in the least average number of clicks to reach a certain IOU (90%) over all the images of a held-out set. It is possible that tuning these parameters specifically for each image might yield better results but we did not do this for simplicity and leave it as an exploration for future work.

D Additional results using key adaptation

We provide additional results here (Fig. 2) using key adaptation within the probabilistic attention BoTNet50 architecture employing full position embedding: ProbBoTNet50-FullPE (Section 4.1). See Section 4.2 for a discussion of the results.



(a) GrabCut

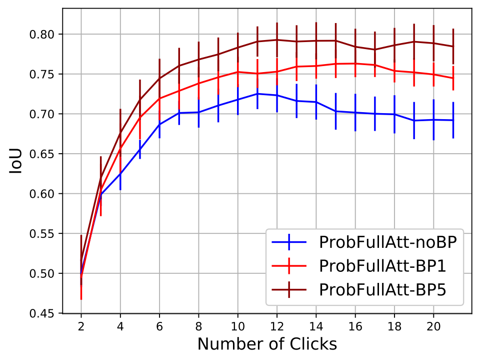


(b) Berkeley

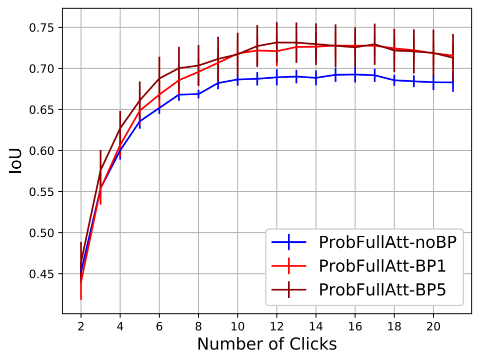
Figure 2: **Unsupervised key adaptation**. Mean IoU vs #clicks with and without key adaptation (KA) on the GrabCut and Berkeley datasets. Probabilistic BoTNets with full position encodings are evaluated without using KA or using 1 iteration of KA with two different prior precision (Prec.) values of 0.001 or 0.

E Additional results using value propagation

We conduct a small scale experiment to demonstrate the effect of value propagation using full attention instead of axial attention. For this experiment, we use a full self attention layer at the output of the network in place of the 1x1 conv classifier and feed in images at a resolution of 64 pixels. The corrective clicks are appended as additional inputs to this layer as described in Section 4.3. The small input resolution allows us to work within the memory limits of current GPUs while being able to use full attention at the output layer [12]. We use the Imagenet classification pre-trained HRNetV2+OCR [14] architecture and fine-tune it on SBD dataset [4] on the interactive segmentation task. Following the same protocol as in Section 4.4, the results are shown in Fig. 3.



(a) GrabCut



(b) Berkeley

Figure 3: **Effect of value propagation using full attention**. Mean IoU vs #clicks with and without value propagation. We use 1 (BP1) and 5 (BP5) iterations of value propagation at the output self attention based classification layer of a network and test on the GrabCut (left) and Berkeley (right) datasets.

F Code

A PyTorch layer implementing our approach is publicly available [1]. It can be imported as any other PyTorch layer as it is subclassed from torch.nn.Module. It implements only the attention update and can be plugged into attention based architectures at the value update operation (e.g. in place of dot-product attention update). Our implementation is parameter free, i.e., it implements only the update and the user is free to define parameters for query, key and value mappings and position embeddings, in any way they choose for their architecture(s). In order to use it for interactive segmentation (e.g. to reproduce results in the paper), one can choose to use any of the publicly available repositories for interactive segmentation, axial attention, etc. and replace the attention updates (with suitable modifications to their architectures) with our attention update.

References

[1] R. Benenson, S. Popov, and V. Ferrari. Large-scale interactive object segmentation with human annotators. In *CVPR*, 2019.

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[3] Agrim Gupta, Piotr Dollár, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation, 2019.

[4] Bharath Hariharan, Pablo Arbelaez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *International Conference on Computer Vision (ICCV)*, 2011.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[1]`https://github.com/apple/ml-probabilistic-attention`

[6] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *Proceedings of the Eighth International Conference on Learning Representations (ICLR 2020)*, April 2020.

[7] Soumajit Majumder and Angela Yao. Content-aware multi-level guidance for interactive instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[8] K.-K. Maninis, S. Caelles, J. Pont-Tuset, and L. Van Gool. Deep extreme cut: From extreme points to object segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[9] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[10] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models, 2019.

[11] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut" – interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, pages 309–314, 2004.

[12] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition, 2021.

[13] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *European Conference on Computer Vision (ECCV)*, 2020.

[14] Y. Yuan, X. Chen, and J. Wang. Object-contextual representations for semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2019.