Supplementary Material for "Amortized Inference with User Simulations"

A TECHNICAL DETAILS

With this section, we provide technical details of the two network structures utilized in the conditional density estimator implemented: *query–key–value cross-attention* (for the encoder network) and *Glow* (for the conditional INN).

A.1 QKV Cross-Attention

To achieve fixed-size behavior features ($\tilde{\boldsymbol{y}}$) from multi-trial observations ($\boldsymbol{y}$) in a permutation-invariant manner, the authors of BayesFlow suggested the use of a network based on sum-pooling of the MLP outputs of each trial observation [1]. Our study implemented an encoder network with the recently introduced QKV cross-attention structure [5, 9], from which we have empirically observed better performance.

QKV cross-attention is a method to extract values from given K–V pairs in accordance with Q. In our setting, each K–V pair is determined for every trial observation in the given observation set ($\boldsymbol{y}$), and Q is a fixed-size vector with learned parameter values. The attention layer measures the *score* of each K per Q (e.g., the dot-product), and it outputs the weighted sum of V in line with the scores of the corresponding K. In consequence, irrespective of the number of K–V pairs (i.e., trials observed), the cross-attention output has the same size as Q (a fixed size). That is, the attention-based encoder network allows the extraction of fixed-size features ($\tilde{\boldsymbol{y}}$) from data of variable size ($\boldsymbol{y}$). Following recent studies' lead [4, 5], we applied cross-attention and self-attention (which together determine all Q, K, and V from the same source, in an architecture often called Transformer [9]) in turn for more efficient feature extraction. This attention-based encoder network is applicable whether or not each trial's observation set takes the form of simple aggregated metrics (e.g., completion time) or time-series data with variable length (e.g., behavioral trajectory). For the former case, a K–V pair for cross-attention can be determined by passing each trial observation to simple MLPs. For the latter case, some neural networks that handle sequential data (e.g., a recurrent neural network or Transformer) can function to determine a K–V pair from the time-series data of each trial.

A.2 Glow

We implemented our conditional INN with recently introduced Glow [6] structures. The model enables invertible transformation based on the affine coupling layer [2]. The affine-coupling-layer-based transformation completes the following processes.

Suppose a bi-directional transformation between $\boldsymbol{u}$ and $\boldsymbol{v}$ when given a feature vector $\tilde{\boldsymbol{y}}$. For forward transformation $\boldsymbol{u} \rightarrow \boldsymbol{v}$, $\boldsymbol{u}$ is split into halves $(\boldsymbol{u}_1, \boldsymbol{u}_2)$; then, the output $\boldsymbol{v}$ can be obtained by concatenation of $(\boldsymbol{v}_1, \boldsymbol{v}_2)$, where

$$\boldsymbol{v}_1 = \boldsymbol{u}_1 \odot \exp(NN_1(\boldsymbol{u}_2, \tilde{\boldsymbol{y}})) + NN_2(\boldsymbol{u}_2, \tilde{\boldsymbol{y}}),$$
$$\boldsymbol{v}_2 = \boldsymbol{u}_2 \odot \exp(NN_3(\boldsymbol{v}_1, \tilde{\boldsymbol{y}})) + NN_4(\boldsymbol{v}_1, \tilde{\boldsymbol{y}}).$$

The operator $\odot$ denotes element-wise multiplication. The transformation requires four nonlinear mapping functions (neural networks), $NN_1 \sim NN_4$. For each nonlinear NN function, $\tilde{\boldsymbol{y}}$ is inserted as an external input to the neural network along with $\boldsymbol{u}_2$ or $\boldsymbol{v}_1$; thus, the transformation is conditioned on given $\tilde{\boldsymbol{y}}$. This $\boldsymbol{u} \rightarrow \boldsymbol{v}$ transformation has the perfect inverse operation $\boldsymbol{v} \rightarrow \boldsymbol{u}$ (likewise conditioned on given $\tilde{\boldsymbol{y}}$). One can inversely obtain $\boldsymbol{u}$ via concatenation of $(\boldsymbol{u}_1, \boldsymbol{u}_2)$,

Table B1. Hyperparameters for DQN training in Case 1.

Parameter name	Value
replay memory size	100000
gamma	0.99
tau	0.001
min. eps	0.05
eps decay	0.99999
grad. clipping	0.5
optimizer	Adam
learning-rate schedule	cosine annealing with warm restarts [3] (max. learning rate 0.001)

where

$$\boldsymbol{u}_2 = (\boldsymbol{v}_2 - NN_4(\boldsymbol{v}_1, \tilde{\boldsymbol{y}})) \odot \exp(-NN_3(\boldsymbol{v}_1, \tilde{\boldsymbol{y}})),$$
$$\boldsymbol{u}_1 = (\boldsymbol{v}_1 - NN_2(\boldsymbol{u}_2, \tilde{\boldsymbol{y}})) \odot \exp(-NN_1(\boldsymbol{u}_2, \tilde{\boldsymbol{y}})).$$

In addition to the *coupling* layer, the flow step of Glow includes an *actnorm* layer, responsible for scaling and biasing each parameter, and a *1 × 1 convolution* layer, which handles permutation of parameter orders. Each layer also supports inverse operation. In summary, one flow step consists of the sequence actnorm–convolution–affine-coupling conditioned on given $\tilde{\boldsymbol{y}}$, and it is cheap to compute the bi-directional conversion between input and output. Using about 5–10 flow (or Glow) steps has been found sufficient to approximate the complex parameter posterior in typical models [8]. Further implementation details are available from the original Glow paper [6].

B CASE 1: MENU SEARCH

This section provides the implementation details for the control policy of the user simulation model and conditional density estimator employed for Case 1 (menu search).

B.1 Simulation Model Details

For the control policy of the menu-search model, we implemented a Q-network that receives the model parameter values, $\boldsymbol{\theta}$ (size = 4) along with the task state (size = 18). The Q-network consisted of three fully connected (FC) layers, the first two of which each included 64 hidden units with non-linear Rectified Linear Unit (ReLU) activation. The final layer's size (i.e., the number of units) was set as the action dimension of the menu-search environment (here, 9). As for the structure's inputs, the task state is entered only for the first layer, whereas the given model parameters are entered for not only the first layer but also the second and the last, by being concatenated into each intermediate output. That is, the policy model was conditioned on the given model parameters by *feature-level concatenation* [7]. The total number of trainable parameters of the policy model was 6.5K. We performed the training by means of a DQN with the hyperparameters listed in Table B1.

B.2 Density-Estimator Details

For the encoder network, we used a simple MLP composed of four FC layers. The given observation ($\boldsymbol{y}$) had a size of 4. Each of the first three layers included 16 units with ReLU activation. This network's final layer output a 32-D feature

Table B2. Hyperparameters for density-estimator training in Case 1.

Parameter name	Value
grad. clipping	1.0
optimizer	Adam
learning-rate schedule	cosine annealing with warm restarts [3] (max. learning rate 0.00001)

Table C1. Hyperparameters for DQN training in Case 2.

Parameter name	Value
replay memory size	500000
gamma	0.99
tau	0.0001
min. eps	0.05
eps decay	0.99999
grad. clipping	1.0
optimizer	Adam
learning-rate schedule	cosine annealing with warm restarts [3] (max. learning rate 0.001)

vector ($\tilde{\boldsymbol{y}}$). For the conditional INN, we used five Glow steps, and the total number of trainable parameters of the density estimator model was 34.5K. The training employed stochastic gradient descents with the hyperparameters in Table B2.

C CASE 2: POINT-AND-CLICK

C.1 Simulation Model Details

For the control policy of the point-and-click model, we implemented a Q-network that receives the model parameter values ($\boldsymbol{\theta}$), with a size of 4, along with the task state, of size 11. The Q-network consisted of three (FC) layers, the first two of which each included 64 hidden units with ReLU activation. The final layer's size (50) was set as the action dimension, or 50. In this case too, the task state was entered only for the first layer, whereas the other inputs (the given model parameters) were entered for the second and last layers through concatenation into each intermediate output (i.e., via feature-level concatenation). In total, there were 8.6K trainable parameters of the policy model. The training applied a DQN with the hyperparameters listed in Table D1.

C.2 Density-Estimator Details

For Case 2's encoder network, we used an attention-based network structure to extract a fixed-sized feature vector ($\tilde{\boldsymbol{y}}$) from a multi-trial observation ($\boldsymbol{y}$). One K–V pair was determined for each trial observation for QKV cross-attention in the encoder network. Here, each trial observation included a vector composed of aggregated metrics, such as trial performance (with a size of 12), and a 2-D array consisting of a time-series trajectory (feature dimensions = 5). For determining the K–V pairs, we processed the aggregated data and trajectory data separately. The aggregated data were entered into three FC layers with hidden units [64, 64, 24] (i.e., the output size was 24). The trajectory data were fed to the *Perceiver* Transformer [5] to be extracted as a vector with a size of 8. After the processing, a 32-D vector was acquired from each trial observation. This vector was fed to the cross-attention layer as K and V (i.e., one K–V pair

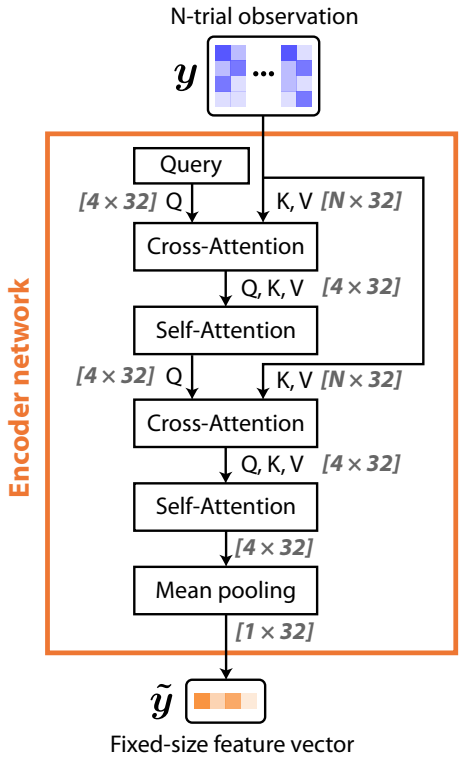

Fig. C1. The encoder network's structure in cases 2 and 3. The size of each intermediate output is indicated in parentheses.

Table C2. Hyperparameters for density-estimator training in Case 2.

Parameter name	**Value**
dropout during attentions	0.4
grad. clipping	0.5
optimizer	Adam
learning-rate schedule	cosine annealing with warm restarts [3] (max. learning rate 0.0001)

per trial). We implemented the Q system as a 4×32 matrix with trainable parameter values. The output of the QKV cross-attention layer was the same as with Q (i.e., a 4×32 matrix); it was fed to an additional self-attention layer. We repeated the cross-attention plus self-attention processes twice (see Figure C1 for the full process within the encoder network and the size of the intermediate outputs).

For the conditional INN, five Glow steps were used. The total number of trainable parameters of the density estimator model was 118K (51.8 K for the encoder network and 66.2K for the conditional INN). We performed the training by means of stochastic gradient descents, using the hyperparameters presented in Table C2.

Table D1. Hyperparameters for PPO training in Case 3.

Parameter name	**Value**
gamma	1.00
entropy coeff.	0.01
value func. coeff.	1.0
grad. clipping	0.2
optimizer	Adam
learning rate	0.00002

Table D2. Hyperparameters for density-estimator training in Case 3.

Parameter name	Value
dropout during attentions	0.2
grad. clipping	1.0
optimizer	Adam
learning-rate schedule	cosine annealing with warm restarts [3] (max. learning rate 0.00005)

D CASE 3: TOUCHSCREEN TYPING

D.1 Simulation Model Details

For the control policy of the touchscreen-typing model, we implemented both actor and critic networks that receive the model parameter values ($\boldsymbol{\theta}$), size = 3, along with the task state, of size 2. Each network consisted of three FC layers, where each of the first two included 64 hidden units with ReLU activation and the final layer's size was set as the action dimension (size = 3). The task state was entered directly into the first layer. The given model parameters too were entered for the first layer but after being passed through an additional FC layer (hidden-unit size = 4). There were 736 trainable parameters of the policy model. The training was performed via PPO with the hyperparameters in Table D1.

D.2 Density-Estimator Details

For the encoder network, we used an attention-based network structure to extract a fixed-sized feature vector ($\tilde{\boldsymbol{y}}$) from a multi-trial observation ($\boldsymbol{y}$). In a contrast against Case 2, each trial observation in Case 3 was only a vector consisting of aggregated metrics (e.g., trial performance) with a size of 5 (i.e., no time-series data were included). Three FC layers with hidden units [16, 16, 32] were used to determine the K–V pair from each trial observation (i.e., a 32-D vector was acquired from each trial observation). In other respects, the encoder-network structure was nearly the same as that implemented for Case 2 (see Figure C1). For the conditional INN, we used five Glow steps. In all, the density-estimator model had 100.9K trainable parameters (68.1K for the encoder network and 32.8K for the conditional INN). The training applied stochastic gradient descents with the hyperparameters in Table D2.

E RESULTS FOR DIFFERENT PRIORS (SUBSECTION 8.2)

Table E1 and Table E2 show the effect of different prior settings on inference performance (parameter-recovery and behavior prediction, respectively) for Case 1. Tables E3 and E4 present the corresponding results for Case 2. We tested the two experiment conditions *Literature-prior* and *Uniform-prior* as described in Subsection 8.2, and both cases yielded results similar to those with Case 3: Firstly, parameter-recovery performance suffered when the trained density estimator

Table E1. Parameter-recovery performance (R^2 values) with the menu-search model (Case 1), with different prior settings. Δ represents the value obtained by subtracting the result of the model learned under *Literature-prior* from the result of that learned under *Uniform-prior*, with the results where $|\Delta| \geq 0.1$ in green or red while those where $0.1 > |\Delta| \geq 0.05$ are colored light green or light red.

Inferred parameter	Evaluated under Uniform-prior			Evaluated under Literature-prior		
	Uniform → Uniform	Literature → Uniform	Δ	Uniform → Literature	Literature → Literature	Δ
d_{fix}	0.897	0.880	0.017	0.742	0.744	−0.002
d_{sel}	0.900	0.874	0.026	0.863	0.871	−0.008
p_{rec}	0.771	0.594	0.177	0.806	0.817	−0.011
p_{sem}	0.330	0.171	0.159	0.316	0.332	−0.016

Table E2. Behavior-prediction accuracy, per distances from actual observations, with different prior settings for the menu-search model (Case 1). For each row, the closest results are in green and the second-closest are in light green.

Behavior	Distance metric	Baseline	Amortized inference	
			Uniform-prior	Literature-prior
Completion time (with target)	Mean diff.	76.791	76.190	43.597
	KLD	0.0741	0.0213	0.0108
	MMD	3.4193	0.4858	0.4787
No. of fixations (with target)	Mean diff.	0.0635	0.0176	0.0018
	KLD	0.8681	0.7903	0.7285
	MMD	0.3850	0.3651	0.3395
Completion time (no target)	Mean diff.	65.972	148.184	186.825
	KLD	0.1007	0.0188	0.0524
	MMD	6.8355	1.0545	0.9976
No. of fixations (no target)	Mean diff.	0.3397	0.2396	0.0798
	KLD	3.9804	1.6368	1.8807
	MMD	0.3603	0.3471	0.1288

Table E3. Parameter-recovery performance (R^2 values) for the point-and-click model (Case 2) with different prior settings, where Δ represents the value obtained by subtracting the result of the model learned under *Literature-prior* from that of the one learned under *Uniform-prior*. The results where $|\Delta| \geq 0.1$ are in green or red, and the results where $0.1 > |\Delta| \geq 0.05$ are in a light green or light red.

Inferred parameter	Evaluated under Uniform-prior			Evaluated under Literature-prior		
	Uniform → Uniform	Literature → Uniform	Δ	Uniform → Literature	Literature → Literature	Δ
σ_v	0.936	0.932	0.004	0.955	0.944	0.011
n_v	0.372	0.420	−0.048	0.294	0.384	−0.090
c_σ	0.622	0.586	0.036	0.446	0.606	−0.160
$T_{h,max}$	0.873	0.692	0.181	0.916	0.925	−0.009

had to cope with data from the other priors. Secondly, *Literature-prior* outperformed using the uniform prior by most behavior-prediction metrics, and *Uniform-prior* produced performance better than each baseline condition's result.

Table E4. Behavior-prediction accuracy with the point-and-click model (Case 2) with different prior settings, expressed via distances from actual observations. For each row, the closest results are in green and the second-closest ones are in light green.

Behavior	Distance metric	Baseline	Amortized inference	
			Uniform-prior	Literature-prior
Completion time	Mean diff.	0.0110	0.0340	0.0433
	KLD	6.5330	5.1014	5.3777
	MMD	0.0868	0.1083	0.1041
Click endpoint (normalized)	Mean diff.	0.5189	0.2936	0.0021
	KLD	0.3568	0.2359	0.2012
	MMD	0.0334	0.0145	0.0072
Cursor travel distance	Mean diff.	0.0339	0.0323	0.0302
	KLD	16.874	15.108	13.608
	MMD	0.0302	0.0257	0.0252

Table E5. Behavior-prediction accuracy with different prior settings, measured via distance from actual observations) with the touchscreen-typing model (Case 3), with strong priors for parameter inference being associated with more accurate predictions for empirical data. For each row, the closest results are in green and the second-closest ones are in light green.

Behavior	Distance metric	Baseline	Amortized inference	
			Uniform-prior	Literature-prior
WPM	Mean diff.	6.8432	1.0698	1.7769
	KLD	1.8028	0.2371	0.1787
	MMD	1.0835	0.0632	0.0376
Error rate	Mean diff.	0.9548	0.6336	0.2327
	KLD	1.5111	0.7758	0.4968
	MMD	1.3805	1.3112	1.2410
Backspace count	Mean diff.	0.8204	1.9484	2.0648
	KLD	0.1328	0.0944	0.1059
	MMD	0.1202	0.1819	0.2295
KSPC	Mean diff.	0.0865	0.0339	0.0098
	KLD	3.5048	1.6208	0.9203
	MMD	0.4665	0.4373	0.4007

F RESULTS FOR POINT VS. DENSITY ESTIMATORS (SUBSECTION 8.3)

Table F1 and Table F2 present a comparison between the point and density estimator with regard to inference performance (parameter recovery and behavior prediction, respectively) for Case 1. Case 2's corresponding results are presented in tables F3 and F4. For Case 1, the point estimator was implemented by replacing the conditional INN with an MLP having two hidden layers of 512 units each. We implemented the point estimator for Case 2 by using an MLP with two hidden layers, each with 256 units. In both cases, the results were similar to those in Case 3: after fitting of the estimators to the actual-user dataset at the level of individuals, the two estimator types showed comparable performance, and both outperformed the baseline.

Table F1. Parameter-recovery performance (R^2 values) with the two estimator types for the menu-search model (Case 1). Δ represents the value obtained by subtracting the result of the density estimator from the result of the point estimator; there were no results where $|\Delta| \geq 0.05$.

Inferred parameter	Point estimator	Density estimator	Δ
d_{fix}	0.741	0.744	−0.003
d_{sel}	0.863	0.871	−0.008
p_{rec}	0.832	0.817	0.015
p_{sem}	0.364	0.332	0.032

Table F2. Behavior-prediction accuracy with the two estimator types for the menu-search model (Case 1). For each row, the values for the results closest to the actual observations are in green and those second-closest are in light green.

Behavior	Distance metric	Baseline	Amortized inference	
			Point estimator	Density estimator
Completion time (with target)	Mean diff.	76.791	113.329	43.597
	KLD	0.0741	0.0024	0.0108
	MMD	3.4193	0.4996	0.4787
No. of fixations (with target)	Mean diff.	0.0635	0.0744	0.0018
	KLD	0.8681	0.2952	0.7285
	MMD	0.3850	0.2046	0.3395
Completion time (no target)	Mean diff.	65.972	239.335	186.825
	KLD	0.1007	0.0323	0.0524
	MMD	6.8355	1.0053	0.9976
No. of fixations (no target)	Mean diff.	0.3397	0.0553	0.0798
	KLD	3.9804	1.0458	1.8807
	MMD	0.3603	0.1350	0.1288

Table F3. Parameter-recovery performance (R^2 values) for the point-and-click model (Case 2) with the two estimator types. Δ represents the value obtained by subtracting the result of the density estimator from that of the point estimator; the results where $|\Delta| \geq 0.1$ are in green.

Inferred parameter	Point estimator	Density estimator	Δ
σ_v	0.957	0.944	0.013
n_v	0.653	0.384	0.269
c_σ	0.773	0.666	0.107
$T_{h,max}$	0.933	0.955	−0.022

REFERENCES

[1] Benjamin Bloem-Reddy and Yee Whye Teh. 2020. Probabilistic symmetries and invariant neural networks. *Journal of Machine Learning Research* 21 (2020).

[2] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2017. Density estimation using Real NVP. In *International Conference on Learning Representations*.

Table F4. Behavior-prediction accuracy with the point-and-click model (Case 2) with the two estimator types, per distances from actual observations. For each row, the closest results are in green and the second-closest are in light green.

Behavior	Distance metric	Baseline	Amortized inference	
			Point estimator	Density estimator
Completion time	Mean diff.	0.0110	0.0105	0.0433
	KLD	6.5330	6.7493	5.3777
	MMD	0.0868	0.0806	0.1041
Click endpoint (normalized)	Mean diff.	0.5189	0.0364	0.0021
	KLD	0.3568	0.1451	0.2012
	MMD	0.0334	0.0001	0.0072
Cursor travel distance	Mean diff.	0.0339	0.0260	0.0302
	KLD	16.874	13.956	13.608
	MMD	0.0302	0.0188	0.0252

Table F5. The two estimator types' behavior-prediction accuracy (in terms of distance from actual observations) with the touchscreen-typing model (Case 3), with the results attesting to comparable prediction accuracy for empirical data. For each row, the closest results are in green and the second-closest results are shown in light green.

Behavior	Distance metric	Baseline	Amortized inference	
			Point estimator	Density estimator
WPM	Mean diff.	6.8432	1.5309	1.7769
	KLD	1.8028	0.1630	0.1787
	MMD	1.0835	0.0268	0.0376
Error rate	Mean diff.	0.9548	0.6687	0.2327
	KLD	1.5111	0.6015	0.4968
	MMD	1.3805	1.1446	1.2410
Backspace count	Mean diff.	0.8204	3.4858	2.0648
	KLD	0.1328	0.0791	0.1059
	MMD	0.1202	0.0299	0.2295
KSPC	Mean diff.	0.0865	0.0569	0.0098
	KLD	3.5048	2.1101	0.9203
	MMD	0.4665	0.3699	0.4007

[3] Manuel Glöckler, Michael Deistler, and Jakob H Macke. 2017. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*.

[4] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. 2021. Perceiver IO: A general architecture for structured inputs & outputs. In *International Conference on Learning Representations*.

[5] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. 2021. Perceiver: General perception with iterative attention. In *International Conference on Machine Learning*. PMLR, 4651–4664.

[6] Durk P Kingma and Prafulla Dhariwal. 2018. Glow: Generative flow with invertible 1×1 convolutions. *Advances in Neural Information Processing Systems* 31 (2018).

[7] Hee-Seung Moon, Seungwon Do, Wonjae Kim, Jiwon Seo, Minsuk Chang, and Byungjoo Lee. 2022. Speeding up inference with user simulators through policy modulation. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 1–21.

[8] Stefan T Radev, Ulf K Mertens, Andreas Voss, Lynton Ardizzone, and Ullrich Köthe. 2020. BayesFlow: Learning complex stochastic models with invertible neural networks. *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* 30 (2017).