Learning to learn generative programs with Memoised Wake-Sleep Supplementary Material

A MEMORY IN PERCEPTUAL RECOGNITION

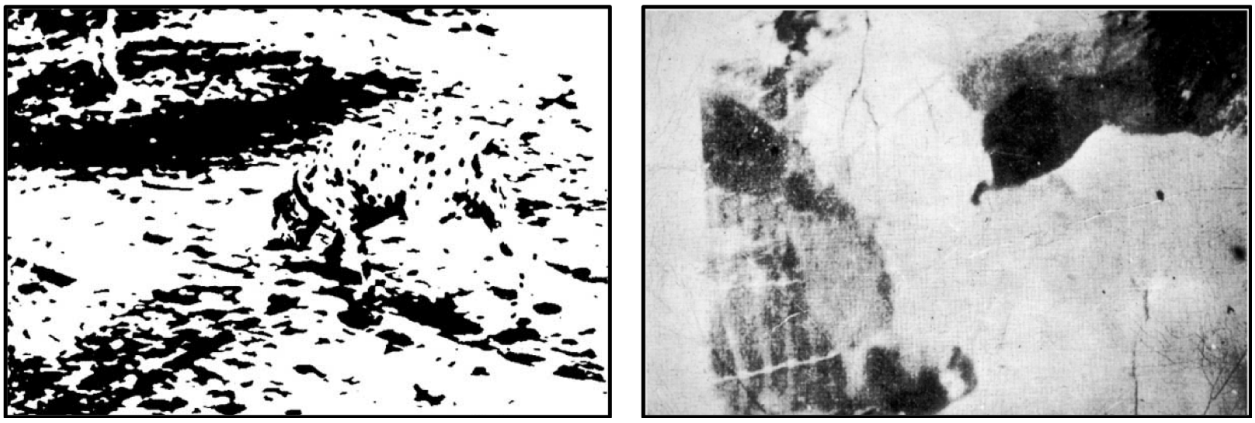

Figure 10: **What is depicted in these images?** Viewers typically find it difficult to see the subject of these images on first presentation, suggesting that it involves solving a very difficult search problem. However, after initial recognition, this structure is immediately apparent on all future viewings, as though the result of previous inference has been stored for future reuse. Answer in footnote

B MWS-FANTASY AND RWS-SLEEP

We present additional results for both MWS and RWS, when the recognition model is trained using prior samples.

B.1 STRING CONCEPTS EXPERIMENT

	$K = 2$	3	5	10
RWS	87.1	86.5	85.2	85.0
RWS-*sleep*	88.9	87.5	86.7	85.5
MWS	84.1	83.1	82.8	82.5
MWS-*fantasy*	82.5	82.7	82.8	82.8

Table 4: Marginal NLL

B.2 CELLULAR AUTOMATA EXPERIMENT

	Easy ($d = 3$)				Hard ($d = 5$)			
	$K = 2$	3	5	10	$K = 2$	3	5	10
RWS	0.74	0.51	0.17	0.09	10.27	5.55	3.44	2.43
RWS-*sleep*	3.39	3.64	2.84	3.12	23.07	20.54	18.05	15.53
MWS	0.01	0.01	0.02	0.02	1.24	1.15	0.90	0.75
MWS-*fantasy*	0.01	0.01	0.01	0.01	5.98	5.12	4.80	4.24

Table 5: Distance from true model

[2]Left: a dalmatian sniffs the ground, (Gregory, 1970). Right: a cow looks towards the camera, (Dallenbach, 1951).

C STRING CONCEPTS DATASET

Our String-Concepts dataset was collected by crawling public GitHub repositories for files with the *.csv* datatype. The data was then automatically processed to remove columns that contain only numbers, English words longer than 5 characters, or common names, and so that each column contained at least 10 elements. We then drew one random column from each file, while ensuring that no more than three columns were included with the same column header. This allows us to reduce homogeneity (e.g. a large proportion of columns had the header 'state') while preserving some of the true variation (e.g. different formats of 'date'). The final dataset contains 5 training examples and 5 test examples for each of 1500 concepts. A sample is included below.

'#574c1d', '#603a0d', '#926627', '#9e662d', '#9e8952'

'(206) 221-2205', '(206) 221-2252', '(206) 393-1513', '(206) 393-1973', '(206) 882-1281'

'ca', 'ja', 'rp', 'tw', 'vm'

'EH15 2AT', 'EH17 8HP', 'EH20 9DU', 'EH48 4HH', 'EH54 6P'

'Exi0000027', 'Exi0000217', 'Exi0000242', 'Exi0000244', 'Exi0000250'

'GDPA178', 'GDPA223', 'GDPA289', 'GDPA428', 'GDPA632'

'YE Dec 11', 'YE Dec 14', 'YE Mar 11', 'YE Sep 08', 'YE Sep 12'

'$0', '$330,000', '$35,720,000', '$4,505,000', '$42,095,000'

'ODI # 2672', 'ODI # 2750', 'ODI # 3294', 'ODI # 3372', 'ODI # 3439'

'3000-3999', '4000-4999', '5000-5999', '50000-59999', 'NA'

'soc135', 'soc138', 'soc144', 'soc67', 'soc72'

'9EFLFN31', 'FE87SA-9', 'LD7B0U27A1', 'PPB178', 'TL88008'

'+1 212 255 7065', '+1 212 431 9303', '+1 212 477 1023', '+1 212 693 0588', '+1 212 693 1400'

'CH2A', 'CH64', 'CH72', 'CH76', 'CH79A'

'20140602-2346-00', '20140603-1148-01', '20140603-1929-04', 'O0601014802', 'O0603155802'

'BUS M 277', 'BUS M 440', 'BUS M 490R F', 'BUS M 490R TTh', 'BUS M 498'

'-2.9065552', '-3.193863', '-3.356659', '-4.304764', '-4.5729218'

'#101', '#4/2/95/2', '#79', '#8/110/3-2', '#94/2'

"'1322563'", "'151792'", "'2853979'", "'5273420'", "'7470563'"

'F150009124', 'F150009169', 'F150009180', 'F150009181', 'F150009346'

'BA-CMNPR2', 'BCOUNS', 'JBGD', 'JDAE', 'OBSB51413'

'b_1', 'e_1', 'g_2', 'k_1', 'o_1'

'P.AC.010.999', 'P.IH.040.999', 'P.IH.240.029', 'P.PC.030.000', 'P.PC.290.999'

'-00:16:05.9', '-00:19:52.9', '-00:24:25.0', '-00:33:24.7', '-00:44:02.3'

'APT', 'FUN', 'JAK', 'KEX', 'NAP'

'SC_L1_O3', 'SC_L3_O2', 'SC_L5_O3', 'SC_L6_O1', 'SC_L6_O2'

'onsen_20', 'onsen_44', 'onsen_79', 'onsen_80', 'onsen_86'

'SDP-00-005', 'SDP-02-106', 'SDP-04-079', 'SDP-06-067', 'SDP-08-045'

'FM0001', 'FM0225', 'FM2500', 'SL0304', 'SS0339'

'BEL', 'KOR', 'PAR', 'POL', 'RUS'

'-0.5423', '-0.702', '0.2023', '0.6124', '0.6757'

'R0353226', 'R0356653', 'R0397240', 'R0474859', 'R0488595'

'CB', 'NA', 'SC', 'SE', 'WE'

'|S127', '|S23', '|S3', '|S4', '|S5'

'GO:0008238', 'GO:0009259', 'GO:0009896', 'GO:0034332', 'GO:0043270'

'MN', 'MO', 'NE', 'SD', 'WY'

'F1-D0N343656', 'F1-D0N343666', 'F1-D0N343669', 'F1-D0N343677', 'F1-D0N343680'

'YATN', 'YBTR', 'YEJI', 'YMGT', 'YPIR'

'ABF', 'AF', 'CBA', 'CC', 'EAJ'

'E', 'NNE', 'NNW', 'W', 'WSW'

'A', 'F', 'G', 'Q', 'R'

'bio11', 'bio14', 'bio16', 'bio19', 'tmin4'

'ACT-B06', 'MS-ACT-C15', 'MS3-08', 'MS960931', 'MS960961'

D CONVERGENCE

In this section, we describe the limiting behaviour of the MWS algorithm, assuming a sufficiently expressive recognition network $r(z|x)$.

The MWS(-replay) algorithm advocated in the paper permits a strong convergence result as long as the encoding probability of each program z is bounded ($r(z|x) > \epsilon_z > 0$) for the top-M programs in the posterior $p(z|x)$. This ensures the optimal zs are eventually proposed. While it is simple to enforce such a constraint in the network structure, we have found this unnecessary in practice.

We consider convergence of r and Q in three cases, providing brief proofs below:

1. **Fixed $p(z, x)$; large memory ($M \to \infty$)**

   The encoder $r(z|x)$ converges to the true posterior $p(z|x)$

2. **Fixed $p(z, x)$; finite memory**

   $r(z|x)$ and $Q_i(z)$ converge to the best M-support posterior approximation of $p(z|x)$.

   $r(z|x)$ will therefore be accurate if $p(z|x)$ if sufficiently sparse, with M elements of $p(z|x)$ covering much of the probability mass. We believe this condition is often realistic for program induction (including models explored in our paper) and it is assumed by previous work (e.g. Ellis et al. 2018).

3. **Learned $p(z, x)$**

   p and Q converge to a local optimum of the ELBO, and $r(z|x)$ matches $Q_i(z)$. This is a stronger result than exists for (R)WS: in (R)WS, optima of the ELBO are fixed points, but convergence is not guaranteed due to differing objectives for p and q (although this is rarely a problem in practice).

Note that, for the MWS-*fantasy* variant of the algorithm, $r(z|x)$ is trained directly on samples from p, and so necessarily converges to the true posterior $p(z|x)$

Proofs

1. If M is large enough to contain the full domain of z, then $Q_i(z)$ is equivalent to enumerating the full posterior. $r(z|x)$ is trained on samples from $Q_i(z)$.

2. Q is updated (to minimize $D_{KL}(Q||P)$) in discrete steps proposed by r. Assuming that r eventually proposes each of the top-M values in $p(z|x)$, these proposals will be accepted and Q converges to the optimal set of programs (with $Q_i(z) \propto p(z|x)$). Then, $r(z|x)$ learns to match $Q_i(z)$.

3. Q and p are optimized to the same objective (ELBO), using SGD for p and in monotonic steps for Q. The algorithm therefore converges to a local optimum of (p, Q), and r converges to Q_i.

E SPARSITY

In order to provide empirical support to the theoretically founded suggestion that MWS relies on posterior sparsity, we ran an additional experiment building on the synthetic GMM model described in section 4.1. In this experiment, we vary the cluster variance σ^2 in the dataset, as a way to produce different levels of posterior sparsity. That is, a large σ^2 produces more overlap between clusters, which leads to less certainty over which cluster an observation came from (less posterior sparsity).

Below, we show the improvement in posterior accuracy gained by using MWS (compared to VIMCO/RWS) for a range of cluster variances σ^2 and particles K. We find significant improvement from MWS for low σ^2 (σ^2=0.03, as in Figure 4) which diminishes as σ^2 increases.

σ^2	0.03	0.1	0.3	1.0
K	Improvement in D_{KL} by MWS			
2	10.26	3.35	0.99	0.07
5	10.21	3.57	0.73	-0.06
10	6.62	3.13	1.35	0.77
20	5.42	3.25	2.24	2.13

Table 6: The D_{KL} improvement is defined as: $\min\Big(D_{KL}(Q_{\text{vimco}}||p), \quad D_{KL}(Q_{\text{rws}}||p)\Big) - D_{KL}(Q_{\text{mws}}||p)$

E.1 MAP INFERENCE

Note, at the lower limit of $M = 1$ (which corresponds to $K = 2$ in the table above), MWS may be seen as simply an algorithm for learning with amortized MAP inference.

This task has also been approached with Deep Directed Generative Autoencoders (Ozair & Bengio, 2014), which are closely related to sparse autoencoders, and may also be seen as a special case of VAEs in which the recognition network $q(z)$ is deterministic.

F HANDWRITTEN CHARACTERS MODEL

We use an LSTM for both the prior $p(z)$ and the recognition model $r(z|x)$, where the output is a sequence of (strokeid, on/off) tuples. In the case of alphabet generalisation, both the prior and recognition model are conditioned on the alphabet index.

Each stroke j in the stroke bank is parametrised by its total displacement ϕ_j^x, ϕ_j^y, and parameters w and u to control the stroke width and sharpness. For the likelihood $p_\phi(x|z)$ we use a differentiable renderer to draw strokes on a canvas: each pixel in the image is set to a value of

$$\max(0, e^{\phi_j^{u\,2}(\phi_j^{w\,2}-d)})$$

where d is the shortest distance from that pixel to the stroke. During rendering, all strokes are placed end-to-end onto the canvas and we then take the output to be the logits of a Bernoulli likelihood.

We include an additional parameter ϕ_j^a to each stroke, which corresponds to the *arc angle*, where ϕ_j^x, ϕ_j^y still correspond to the straight-line distance from the start point to the end point. This angle ranges from -2π to 2π with no discontinuity, corresponding to the two possible orientations of the circle (clockwise or anticlockwise).

We learn our model using ADAM, using 10 characters per alphabet as training data.

To evaluate the log marginal likelihood $\log p(x)$, we fix the generative model, and refine the recognition model by training for additional 20k iterations (batch size = 50) with $K = 200$. We use this refined recognition model to estimate $\log p(x)$ using the IWAE bound with 5k samples. We run this procedure with RWS, VIMCO and Sleep-fantasy training and report the best $\log p(x)$.

G TIMESERIES DATA

As a preliminary experiment, we applied MWS to the task of finding explainable models for time-series data. We draw inspiration from Duvenaud et al. (2013), who frame this problem as Gaussian process (GP) kernel learning. They describe a grammar for building kernels compositionally, and demonstrate that inference in this grammar can produce highly interpretable and generalisable descriptions of the structure in a time series.

We follow a similar approach, but depart by learning a set of GP kernels jointly for each in timeseries in a dataset, rather than individually. We start with time series data provided by the UCR Time Series Classification Archive. This dataset contains 1-dimensional times series data from a variety of sources (such as household electricity usage, apparent brightness of stars, and seismometer readings). In this work, we use 1000 time series randomly drawn from this archive, and normalise each to zero mean and unit variance.

For our model, we define the following simple grammar over kernels:

$$K \to K + K \mid K * K \mid \text{WN} \mid \text{SE} \mid \text{Per} \mid \text{C}, \quad \text{where} \tag{3}$$

- WN is the *White Noise* kernel, $K(x_1, x_2) = \sigma^2 \mathbb{I}_{x_1 = x_2}$
- SE is the *Squared Exponential* kernel, $K(x_1, x_2) = \exp(-(x_1 - x_2)^2 / 2l^2)$
- Per is a *Periodic* kernel, $K(x_1, x_2) = \exp(-2\sin^2(\pi|x_1 - x_2|/p)/l^2)$
- C is a *Constant*, c

We wish to learn a prior distribution over both the symbolic structure of a kernel and its continuous variables (σ, l, etc.). To represent the structure of the kernel as z, we use a symbolic kernel 'expression': a string over the characters

$$\{(,), +, *, \text{WN}, \text{SE}, \text{Per}, \text{C}\}$$

We define an LSTM prior $p_\theta(z)$ over these kernel expressions, alongside parametric prior distributions over continuous latent variables ($p_{\theta_\sigma}(\sigma), p_{\theta_l}(l), \ldots$). As in previous work, exact evaluation of the marginal likelihood $p(x|z)$ of a kernel expression z is intractable and so requires an approximation. For this we use a simple variational inference scheme which cycles through coordinate updates to each continuous latent variable (up to 100 steps), and estimates a lowerbound on $p(x|z)$ using 10 samples from the variational distribution.

Examples of latent programs discovered by our model are displayed in Figure 11. These programs describe meaningful compositional structure in the time series data, and can also be used to make highly plausible extrapolations.

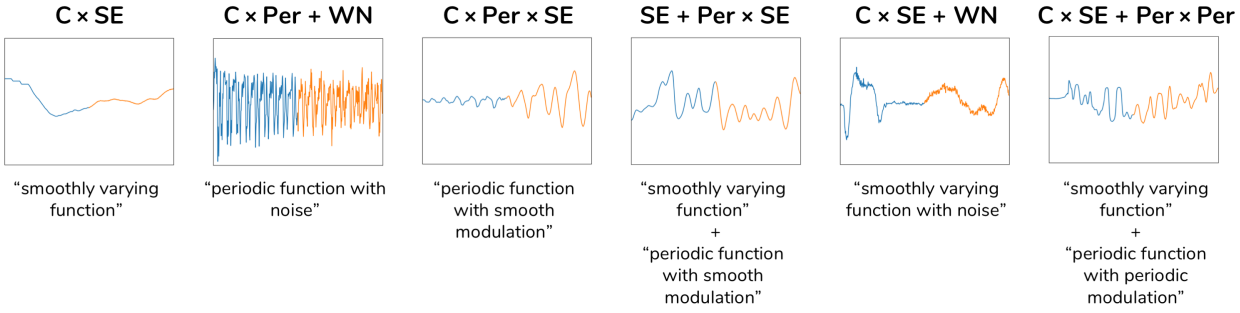

Figure 11: Learning to model the UCR time series dataset with Gaussian Processes, by inferring a latent kernel z for each observed timeseries. Blue (left) is a 256-timepoint observation x_i, and orange (right) is a sampled extrapolation using the inferred kernel $z \sim Q_i$ (symbolic representation above, natural language representation below). During learning, we use Q_i as a memory for the discrete structure of kernels, but use a Variational Bayes inner loop to marginalise out a kernel's continuous variables when evaluation of $p(z, x)$ is required.