GBP Learning

Learning in Deep Factor Graphs with Gaussian Belief Propagation

Seth Nabarro[1] Mark van der Wilk[2] Andrew J. Davison[1]

Google DeepMind, Vicarious AI Reading Group

18th December 2023

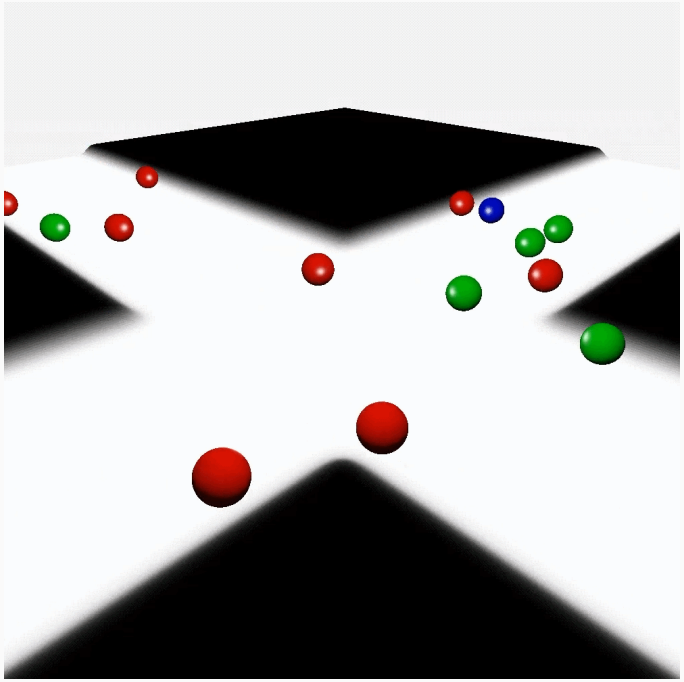
[1]Dyson Robotics Lab, Imperial College London

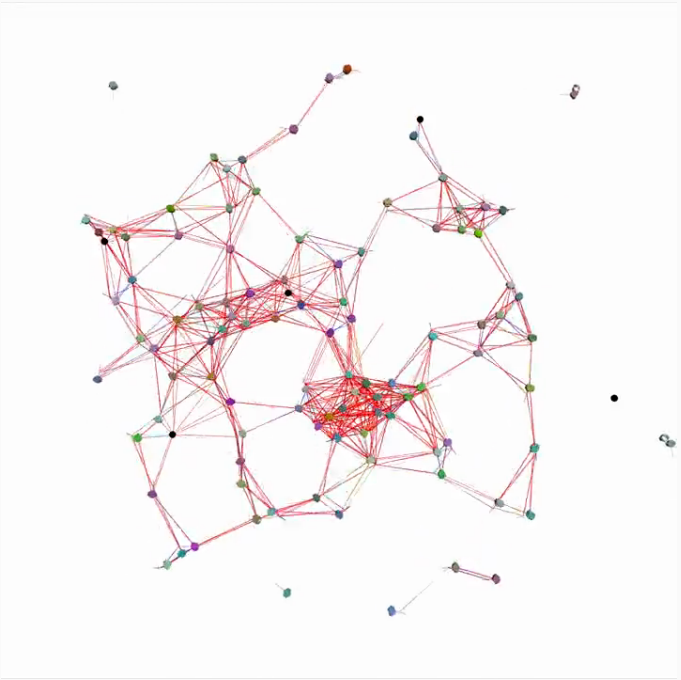
[2]Department of Computer Science, University of Oxford

Gaussian Belief Propagation

GBP has been shown to be a robust distributed solver which works in challenging (nonlinear, non-Gaussian) settings. Recent examples include

- Bundle adjustment and SLAM (Ortiz et al., 2020, 2022)
- Multi-robot localisation (Murai et al., 2022)
- Multi-robot planning (Patwardhan et al., 2022)





Soft-switching in GBP

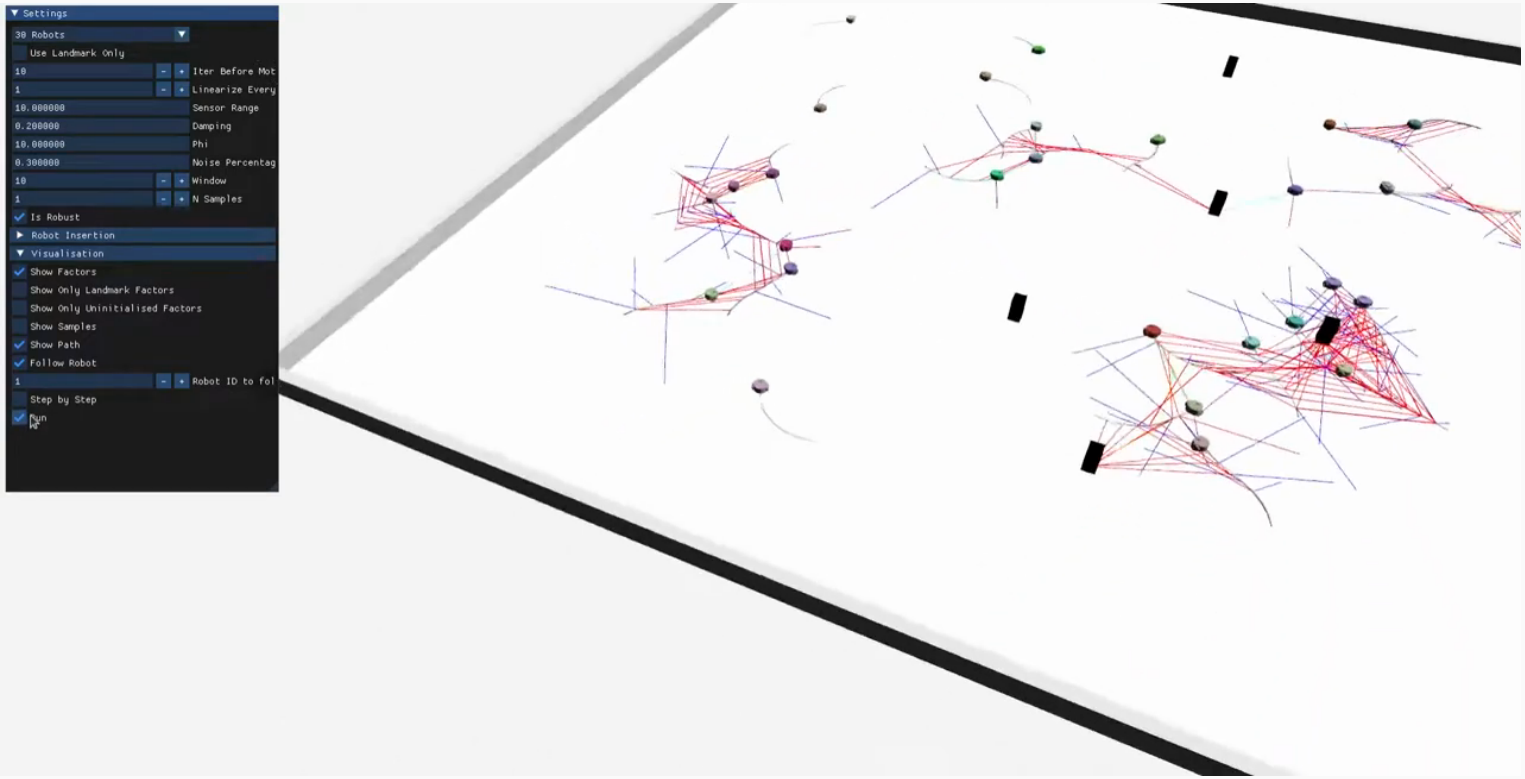


Figure 1: Murai et al. (2022)

Soft-switching in DNNs

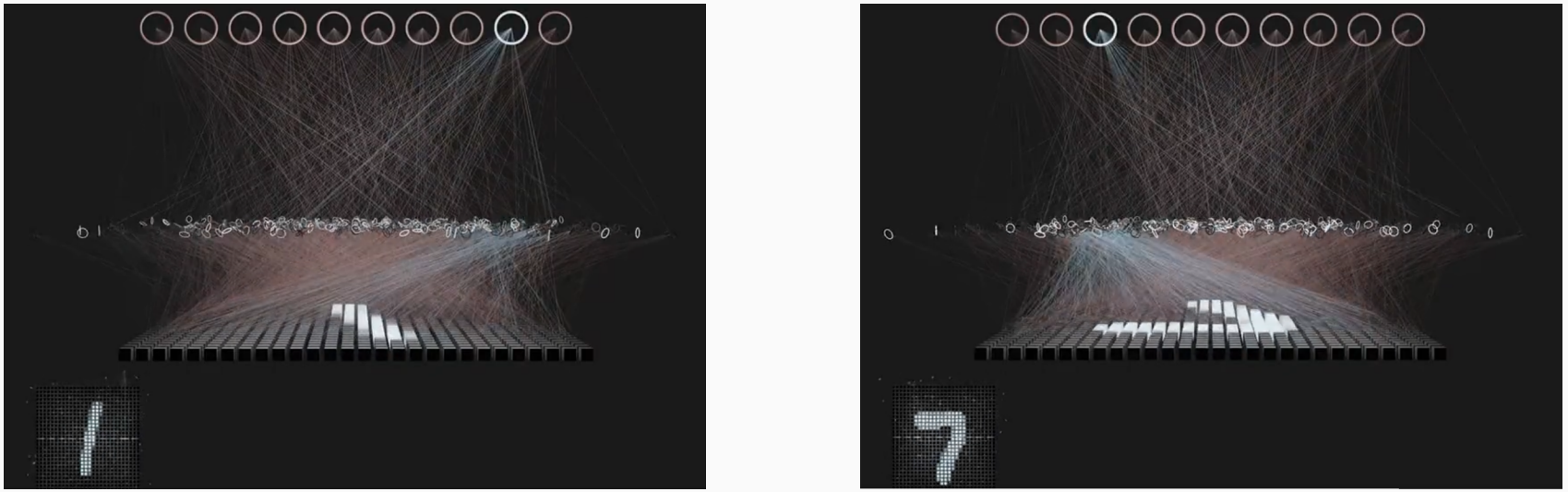


Figure 2: @BenThePearman

Research Questions

1. Can we use GBP to learn model parameters?
2. How should we design learning systems to be trainable in this way?
3. What properties do these systems have?

Flexible Models

Many applications warrant models which can learn

- incrementally and in-the-loop, trading of plasticity and stability on the go,
- without constant supervision signal,
- alongside hand-designed solvers.

Existing DL systems are lacking in these respects.

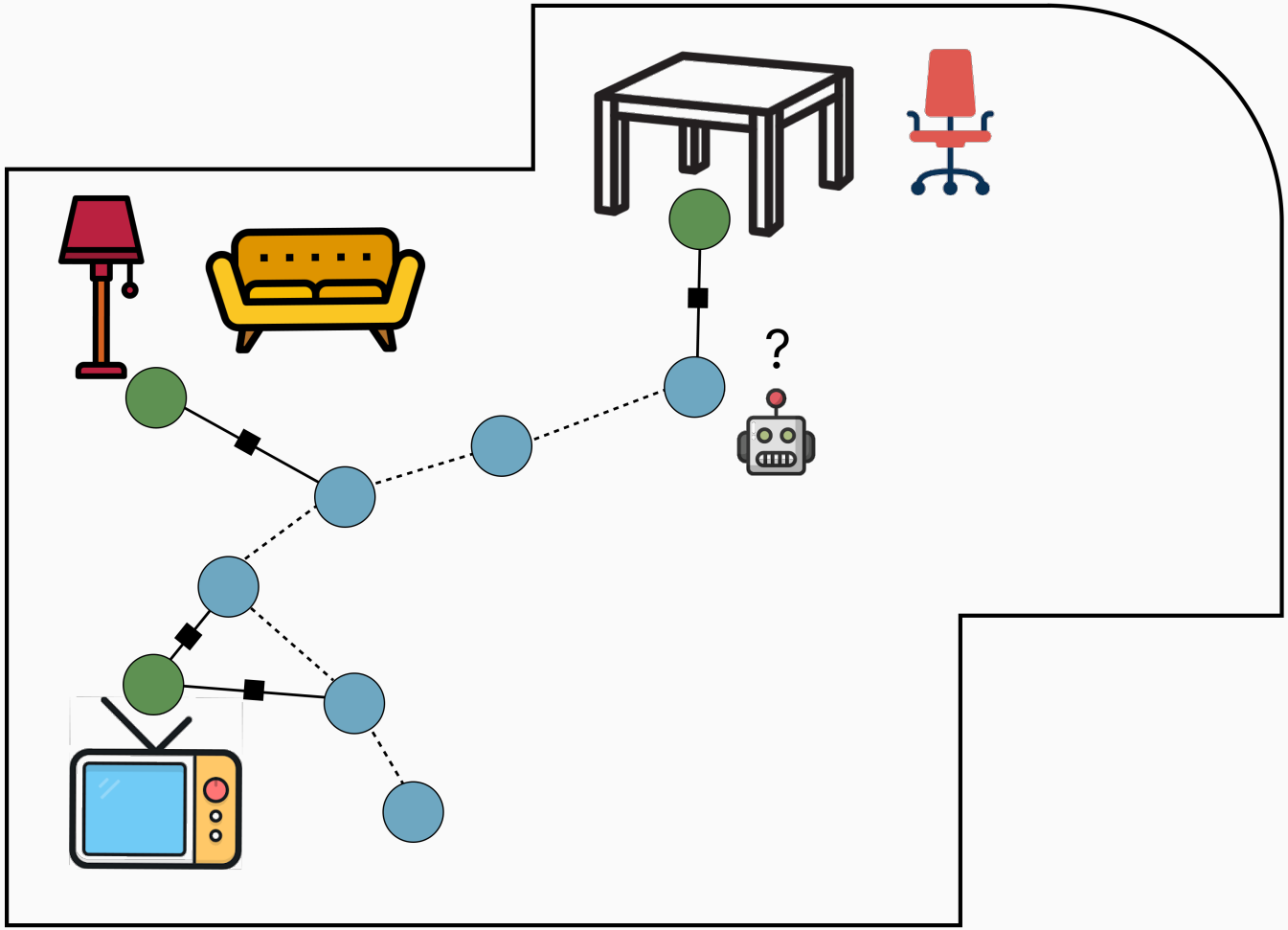
Motivating Example 1: Learning in the Loop

Suppose we have a robot exploring a scene. They can recognise some landmarks in a known map however, the true wheel odometry model is not known. Can we still use the odometry to improve our localisation?

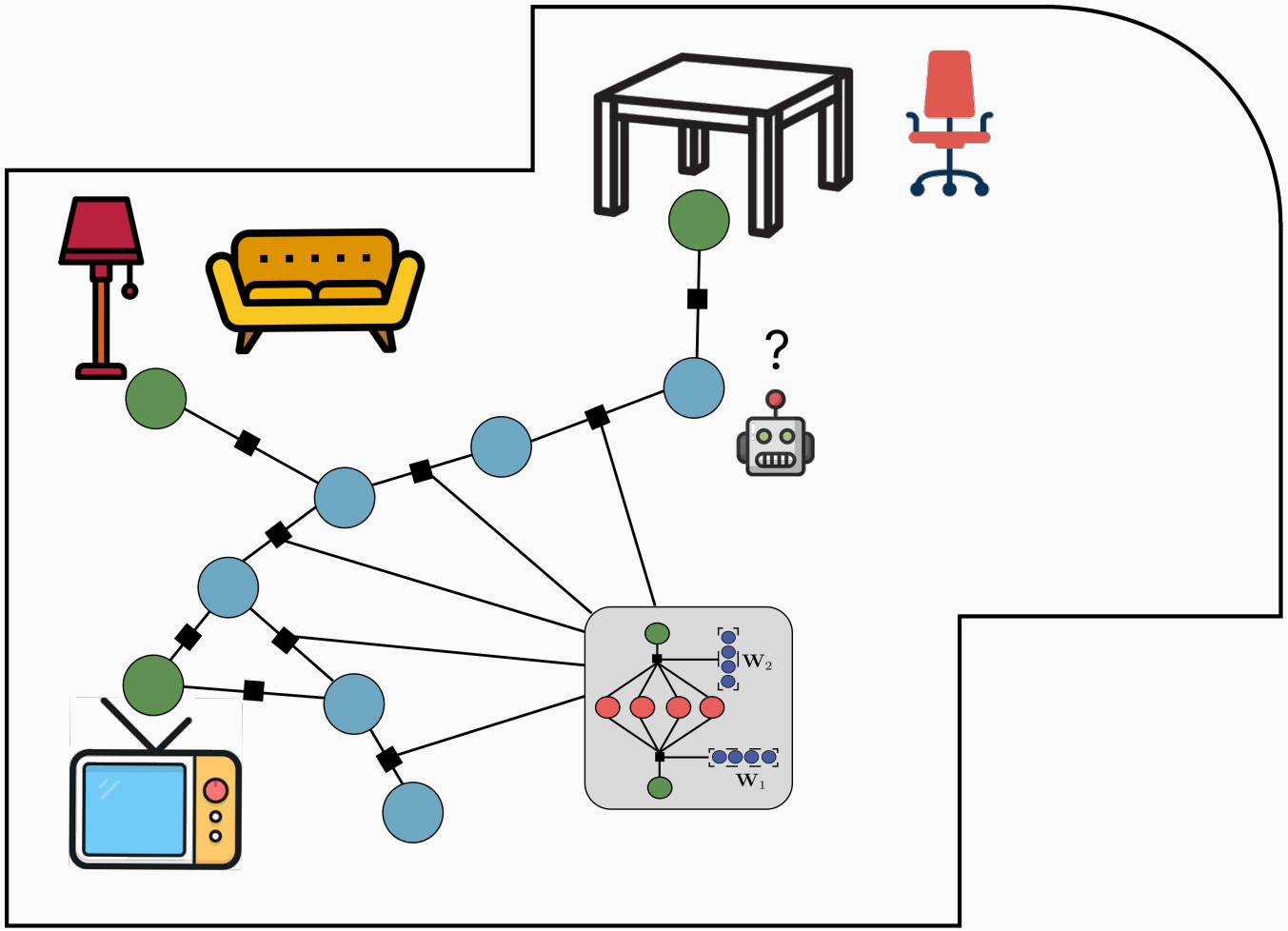
We could:

1. assume an odometry model based on e.g. prior knowledge or physics,
2. collect a dataset and train a model offline, fix the model and do localisation,
3. ignore the wheel odometry, use the visual features only.

Motivating Example 1: Learning in the Loop



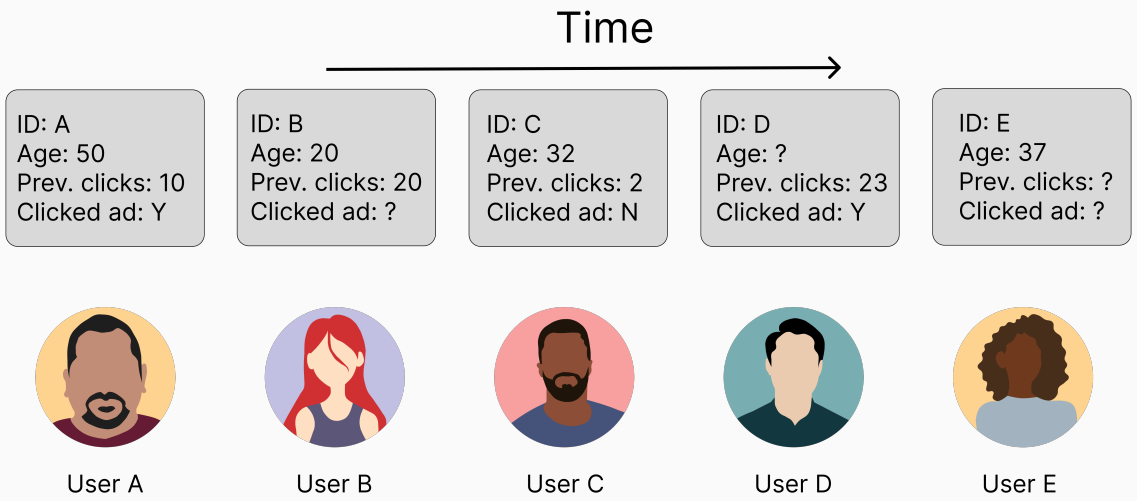
Motivating Example 1: Learning in the Loop



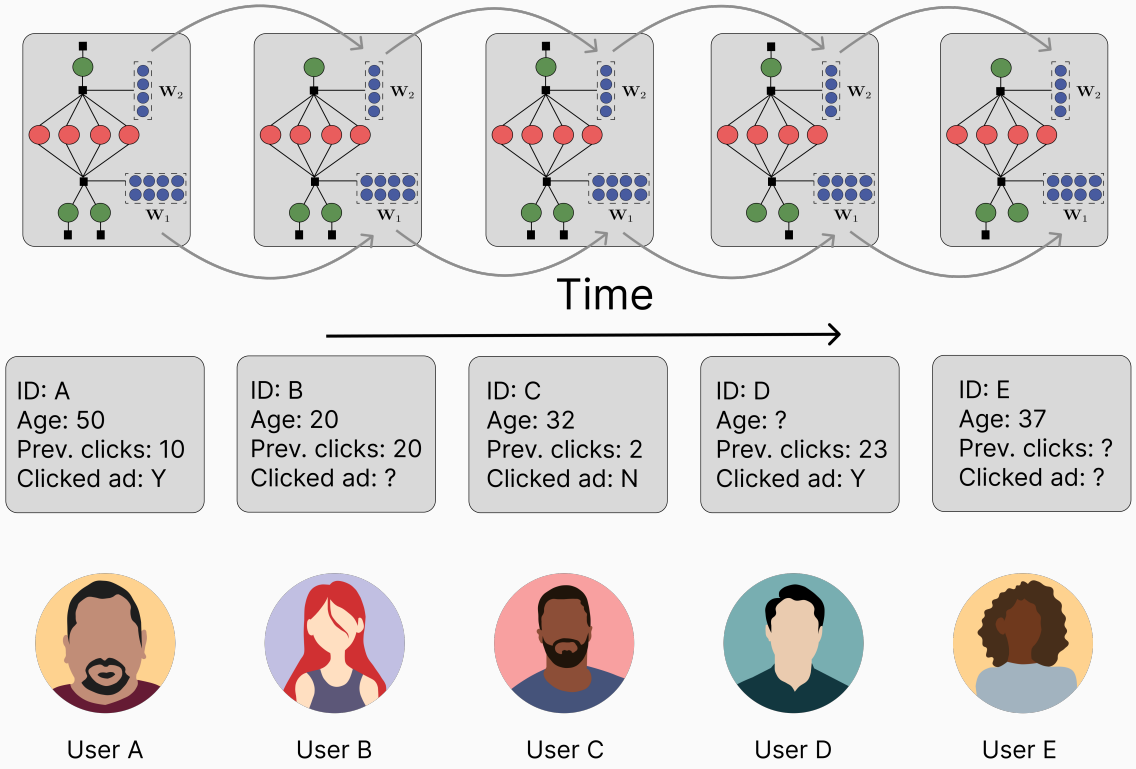
Motivating Example 2: Streaming Data

Suppose we want to learn a user behaviour model in a setting where

1. Data cannot be stored
2. Variables are missing for some observations
3. Not all observations have labels



Motivating Example 2: Streaming Data



Flexible Computation

Backprop is strict about the ordering of operations, which can present challenges for distributed training. Could learning with BP enable more flexible learning systems? Could this better leverage recent hardware platforms with memory colocated to compute?

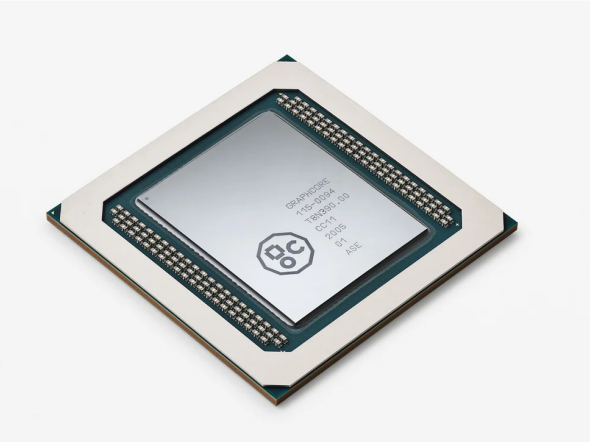


Figure 3: Graphcore IPU

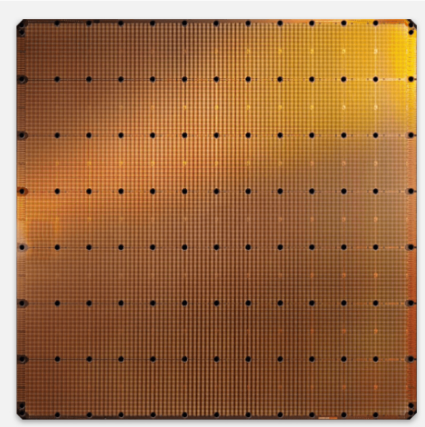


Figure 4: Cerebras WSE-2

Table of contents

Background

Factor graphs

Factor graphs are compact representations of probability distributions. Each factor $\phi_j(\mathbf{x}_j)$ represents the compatibility between different variables. Through inference in such a factor graph, we can find the most likely configuration of variables, given the fixed assumptions encoded in the factors.

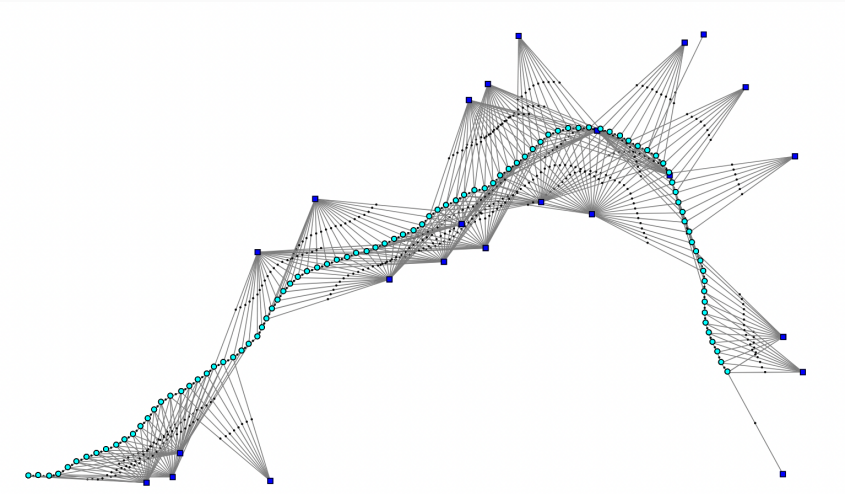


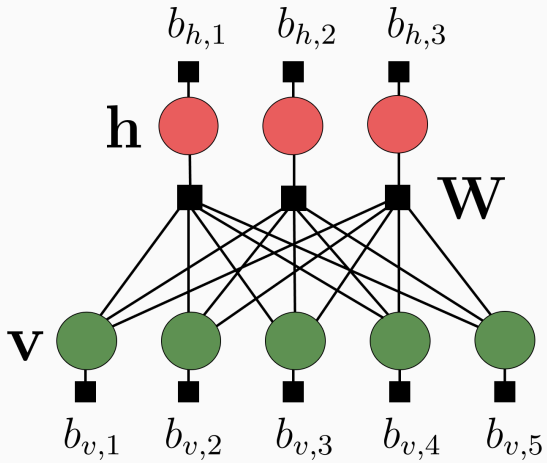
Figure 5: Dellaert et al. (2017)

Learning in Factor Graphs

Traditionally, learning and inference are viewed as two distinct procedures.

- Contrastive divergence
- Expectation maximisation
- Deep learning

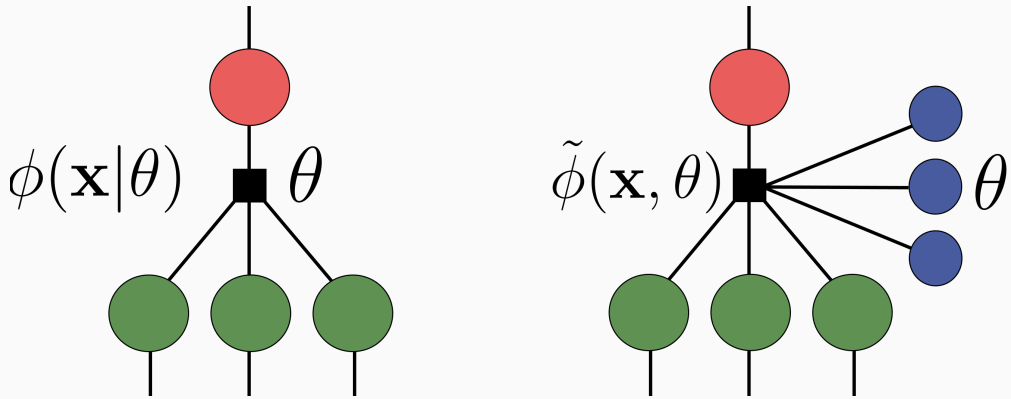
In this setting each factor has some parameters $\phi_j(\mathbf{x}_j) = \phi_j(\mathbf{x}_j|\theta_j)$ which are learned during training, and fixed thereafter.



Learning within inference

Can we do inference and learning as part of the same procedure?

We consider including parameters as additional variables in the graph $\tilde{\mathbf{x}}_j = \{\mathbf{x}_j, \theta_j\}$, $\tilde{\phi}_j(\tilde{\mathbf{x}}_j)$, and extending inference to cover both variables $\{\mathbf{x}_j\}$ and parameters $\{\theta_j\}$.

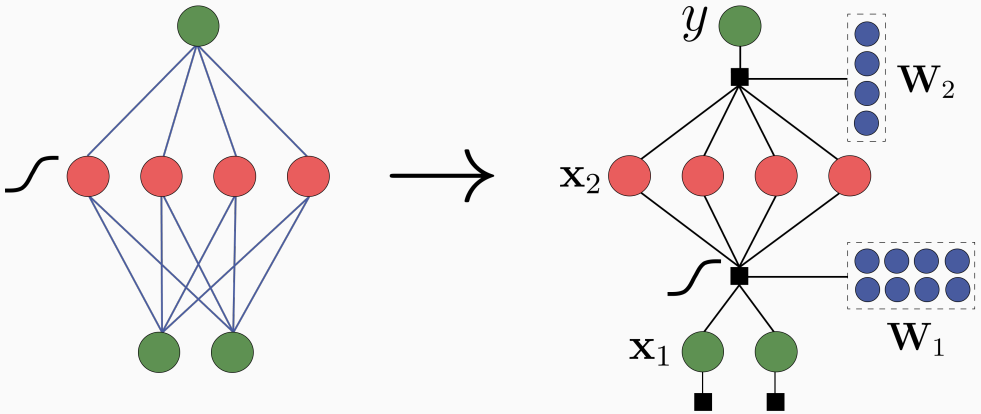


GBP Learning

Model Design

We aim to translate effective architectural designs from DNNs to factor graphs.

Inputs, outputs, activations and parameters are all included as variable nodes. Hidden layers are connected by consistency factors.



Layerwise consistency

The precise form of these factors is a design choice, but in general they either enforce $\mathbf{x}_{l+1} \approx \mathbf{f}_l(\mathbf{x}_l)$ (feedforward) or $\mathbf{x}_l \approx \mathbf{g}_{l+1}(\mathbf{x}_{l+1})$ (generative).

Layerwise Consistency

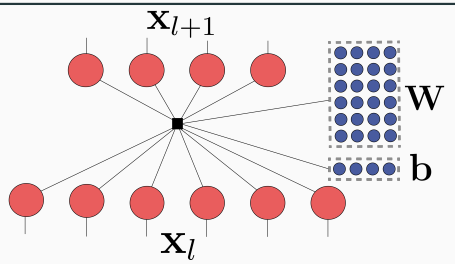
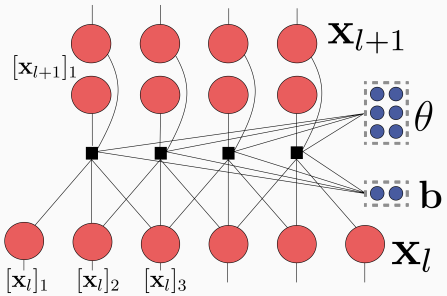
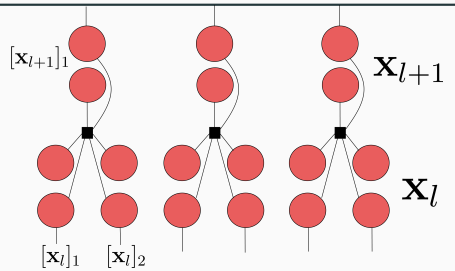
Layer Type	Factor Graph	Factor Energy
Dense		$E_{\text{dense}} = \frac{\left\lvert \mathbf{x}_{l+1} - h\left(\mathbf{W}^T \mathbf{x}_l + \mathbf{b}\right)\right\rvert^2}{\sigma^2}$
Conv.		$E_{\text{conv}} = \frac{\left\lvert [\mathbf{x}_{l+1}]_1 - h\left(\theta^T [\mathbf{x}_l]_{1:3} + \mathbf{b}\right)\right\rvert^2}{\sigma^2}$
Max pool		$E_{\text{mp}} = \frac{\left\lvert [\mathbf{x}_{l+1}]_1 - \max([\mathbf{x}_l]_1, [\mathbf{x}_l]_2)\right\rvert^2}{\sigma^2}$

Table 1: Some examples of inter-layer factors. $h(\cdot)$ is an elementwise activation function.

Inference

As the model is a Gaussian factor graph we can do inference with GBP.

All message update rules are local, so training can be distributed and asynchronous.

GBP in Nonlinear Models

GBP is a solver for linear-Gaussian systems. To apply it to nonlinear factor graphs, we linearise around the current MAP estimate, run a few GBP iterations in the linearised model, relinearise about the updated MAP, etc.

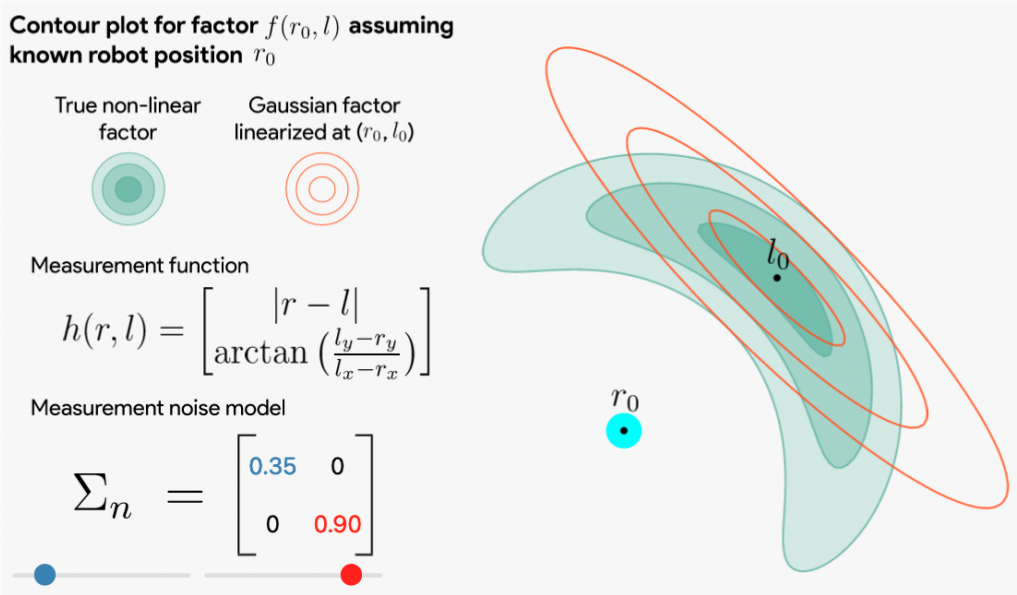


Figure 6: Ortiz et al. (2021)

Efficient Factor to Variable Messages

Naïve computation of the message from a factor j to a connected variable is $O\left(V_j^3\right)$, where V_j is the degree of factor j, making the messages to all variables $O(V_j^4)$.

We reduce this to $O(V_j)$ these updates by

1. Exploiting the low-rank structure of the linearised factor precision matrices via Woodbury identity.
2. One upfront computation $O(V_j)$ to compute intermediates which can be reused for all outgoing messages from a factor.

Continual Learning

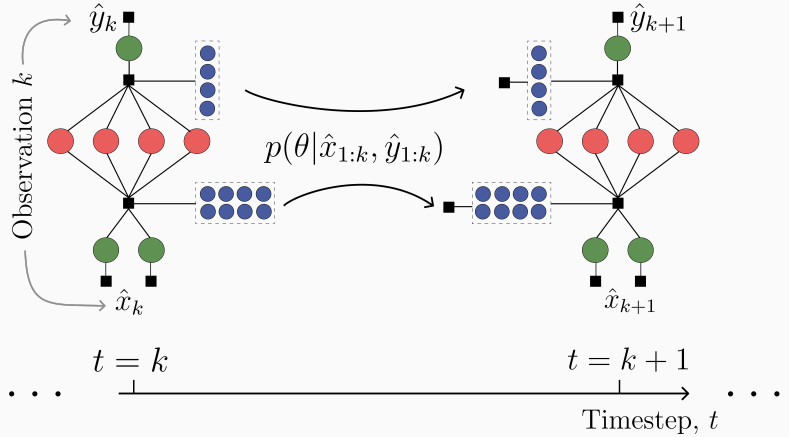
Given the sequence of tasks $[\tau_1, \tau_2, \cdots, \tau_T]$, we learn continually by iterating

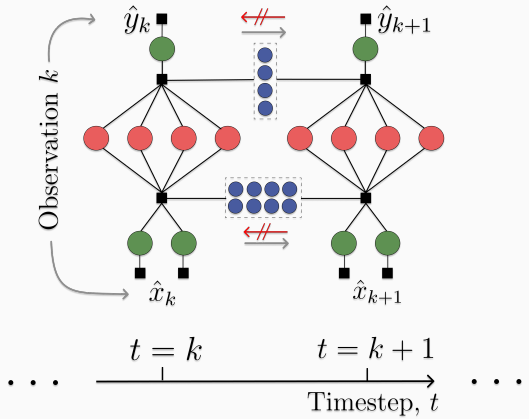
$$\mathcal{D}_k \sim \tau_k \qquad \text{Sample observations for task } k$$

$$p_k(\theta_j) \leftarrow q(\theta_j | \mathcal{D}_{1:k-1}) \qquad \text{Set parameter prior to prev. posterior}$$

$$\{q(\theta_j | \mathcal{D}_{1:k})\}_j \leftarrow \mathrm{GBP}(\mathcal{D}_k, p_k(\theta_j)) \qquad \text{Run GBP in graph for task } k$$

$$k \leftarrow k + 1$$





Why Gaussian?

1. Previous work shows GBP can be effective in the presence of nonlinear relationships and non-Gaussian observations.
2. Guarantees exist for loopy belief propagation in linear-Gaussian models (Weiss and Freeman, 1999; Johnson et al., 2009)
3. Gaussian variables are often a natural representation for (quasi-) continuous signals e.g. pixel intensities
4. No trade off between compute and number of states.

We hope to mitigate the restrictiveness of a Gaussian assumption with overparameterisation + nonlinearities.

Discrete Observations

While we assume the variables in our model are Gaussian distributed, we can use nonlinear factors to model discrete phenomena. For example, we can use Gaussian logits $\mathbf{u}$ to parameterise categorical observations, one-hot y.

$$u_j \sim \mathcal{N}(\mu_j, \sigma_j^2)$$
$$E_{\text{softmax}} = \frac{|y - \text{softmax}(\mathbf{u})|^2}{\sigma^2},$$

and generate class predictions according to

$$p_i \approx \text{softmax}_i(\mathbf{u}) .$$

Implementation

The current codebase is built in `TensorFlow` for GPU acceleration, static graph optimisations, XLA support, `TensorBoard`. Inference is parallelised over a batch.

We note that JAX has similar advantages to `TensorFlow`, though it might also allow for easier multi-device parallelism and be slightly faster.

Related Work

Related Work

There are many interesting connections between our method and other research areas. These can primarily be divided into four areas:

1. Energy-based models
2. Bayesian deep learning
3. Local deep learning
4. Learnable, discrete factor graphs

Related Work: Energy-Based Models

Probabilistic EBMs allocate probability density as

$$p(\mathbf{x}|\theta) = \frac{1}{Z_\theta} \exp\left(-E_\theta(\mathbf{x})\right)$$

where $E_\theta(\cdot)$ is e.g. a DNN which outputs a scalar.

$E_\theta(\cdot)$ can be trained to maximise the likelihood of a dataset $\mathcal{D} := \{\mathbf{x}_i\}$

$$\theta^* = \mathrm{argmax}_\theta \log p(\mathcal{D}|\theta)$$

by contrastive divergence, which approximates the gradients of Z_θ using samples from the model. Samples from $p(\mathbf{x}|\theta)$ are not easy to generate, and are usually drawn by expensive MCMC.

Related Work: Energy-Based Models

Our factor graphs can be viewed as EBMs with quadratic energy functions. As we assume all variables are Gaussian distributed, and we do inference of the posterior marginals, we do not need to worry about normalisation.

As we assume a family of distributions, our model formulation is less flexible than e.g. deep EBMs, however, we seek to counterbalance this with overparameterisation and nonlinear factors.

Related Work: Bayesian Deep Learning

As in Bayesian deep learning (BDL), we seek to infer a posterior distribution over the weights a deep network. Existing approaches to BDL include

1. MCMC (e.g. Zhang et al., 2019; Neal, 2012)
2. (Mean-field) variational inference (e.g. Gal and Ghahramani, 2016; Blundell et al., 2015)
3. Laplace approximation (e.g. MacKay, 1992; Ritter et al., 2018).

In these models, activations are only uncertain due to uncertain parameters. In contrast, our layers are linked by probabilistic factors, so our activations are inherently random. Uncertainty can be resolved by both bottom-up and/or top-down messages.

Related Work: Local Deep Learning

Biological plausibility and computational efficiency have motivated research efforts into alternative approaches to DNN training.

These include *local* deep learning methods which aim to replace the propagation of global loss signal with parameter updates via a local rule or loss. Methods include

- local contrastive learning (Löwe et al., 2019; Hinton, 2022),
- local supervised losses (Belilovsky et al., 2020),
- Hebbian learning (Krotov and Hopfield, 2019).

Belief propagation updates are inherently local and so we not suffer from backward locking. Our models can be trained in a distributed and asynchronous manner.

Related Work: Vicarious i

Query Training (Lázaro-Gredilla et al., 2021) is an approach to learn the parameters θ of a factor graph by framing inference as supervised learning. The algorithm is as follows:

1. A subset of variables in each training example are masked,
2. BP is unrolled given current θ and the observed variables,
3. the inferred marginals in the mask are compared to ground truth and
4. θ updated through backprop.

Related Work: Vicarious ii

Hierarchical compositional feature learning (Lázaro-Gredilla et al., 2016) is similar to ours. Parameters are included as variables in the graph. The main differences with our work is that

- variables in our model are continuous,
- and we do marginal (rather than MAP) inference, allowing continual learning by filtering.

In addition, George et al. (2017) present a factor graph vision model inspired by known connectivities in the visual cortex. MAP inference is executed using BP and some architectural features are learned without supervision based on consistency between neighbouring receptive fields.

Related Work: Training Deep Networks with GBP

IOP Publishing Mach. Learn.: Sci. Technol. 3 (2022) 035005 https://doi.org/10.1088/2632-2153/ac7d3b

MACHINE LEARNING
Science and Technology

CrossMark

PAPER

OPEN ACCESS

RECEIVED
29 January 2022

REVISED
14 April 2022

ACCEPTED FOR PUBLICATION
29 June 2022

PUBLISHED
15 July 2022

Deep learning via message passing algorithms based on belief propagation

Carlo Lucibello[1,*], Fabrizio Pittorino[1], Gabriele Perugini[1,2] and Riccardo Zecchina[1]

[1] Institute for Data Science and Analytics, Bocconi University, Milano, Italy
[2] Department of Applied Science and Technology, Politecnico di Torino, Torino, Italy
* Author to whom any correspondence should be addressed.

E-mail: carlo.lucibello@unibocconi.it

Authors also use GBP to train deep models. They analytically derive the message updates for MLPs with `sign` or `ReLU` activation functions. The resulting updates can be computed more efficiently if the weights and activations are assumed to be discrete. In addition they have to train for multiple epochs and do not consider the potential of BP for distributed training.

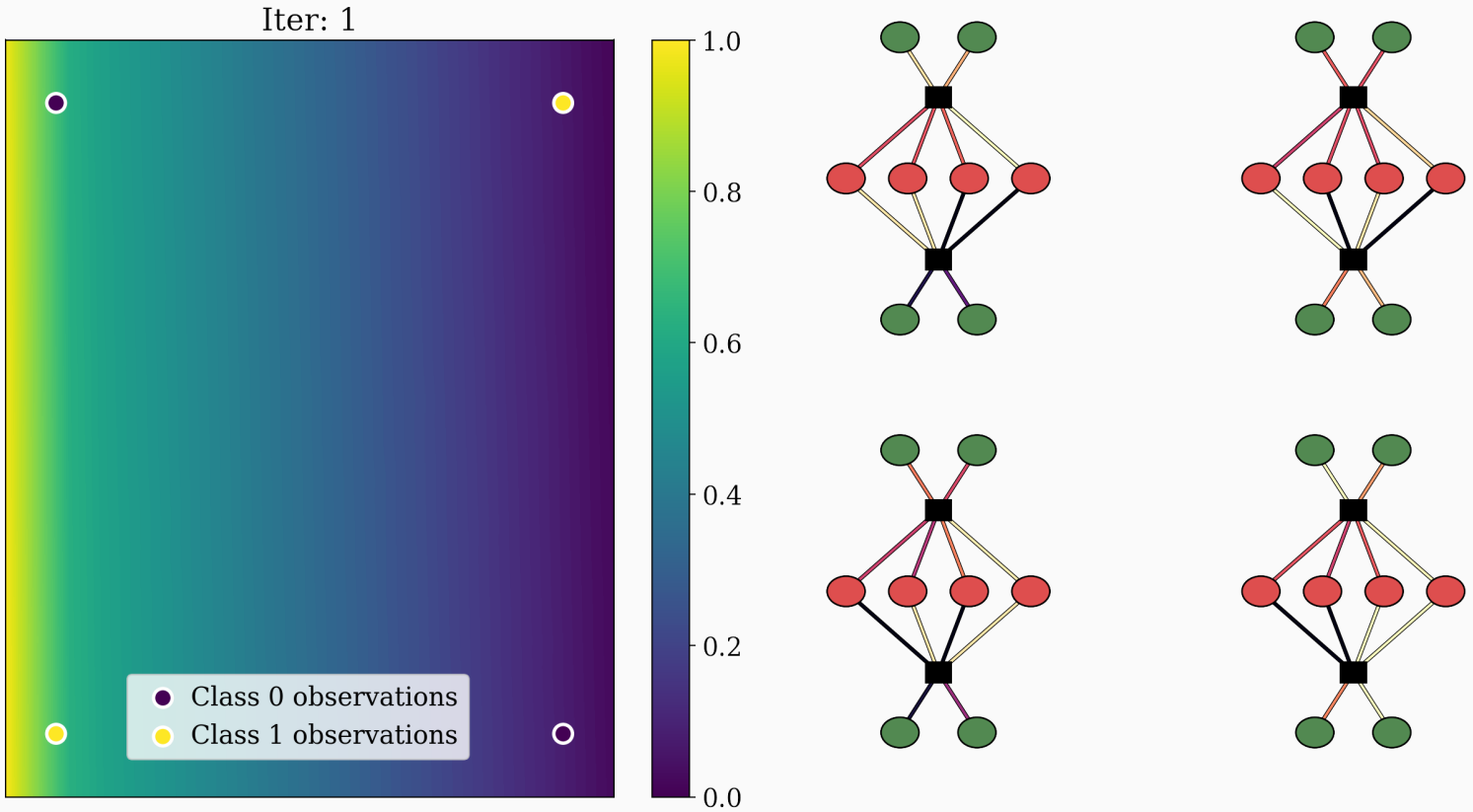
Results

Toy Experiments

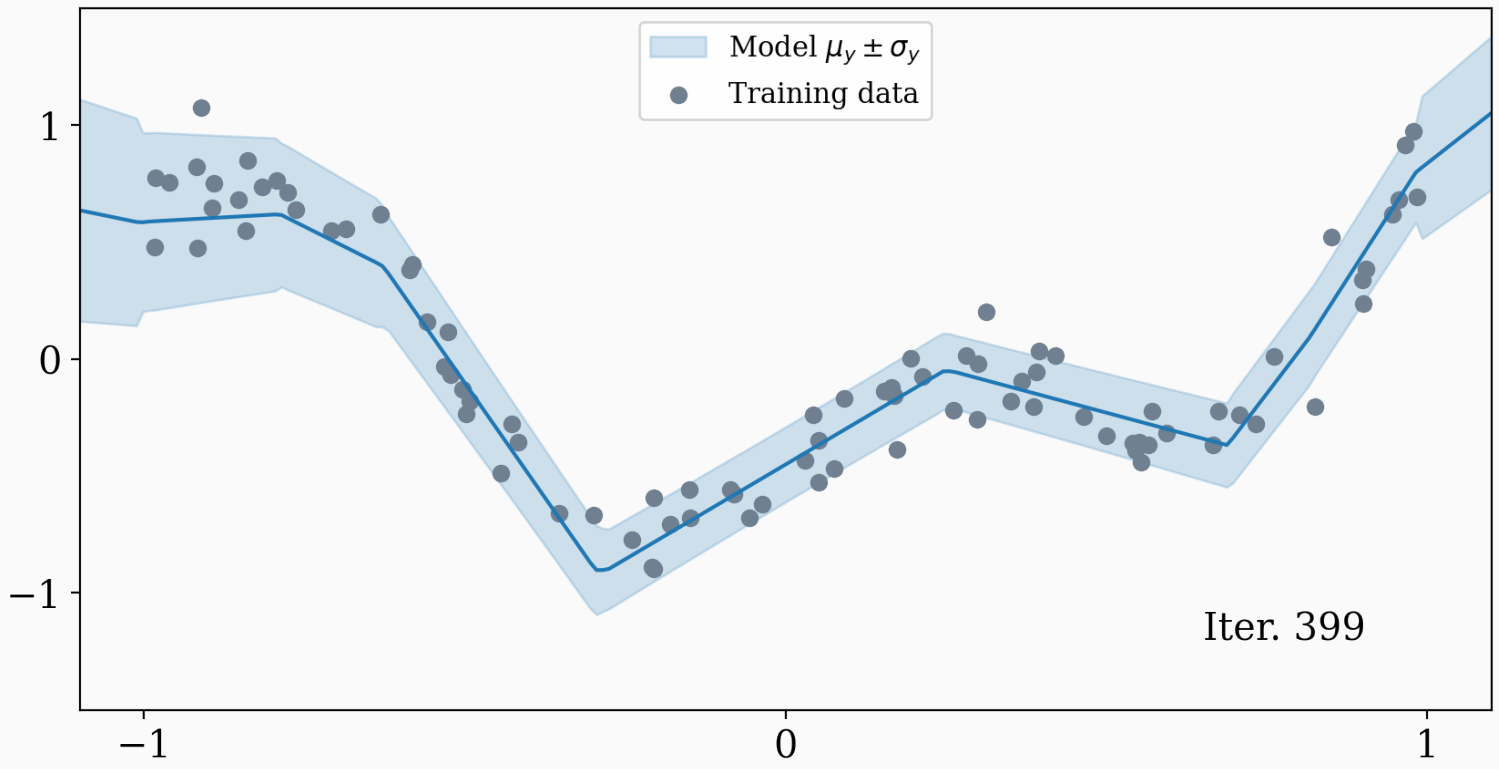
We first evaluate GBP Learning with some MLP-like factor graphs on simple supervised learning tasks.

1. Exclusive-OR classification
2. 1D regression

XOR


Iter: 1
Class 0 observations
Class 1 observations
1.0
0.8
0.6
0.4
0.2
0.0

Regression



Video Denoising

To demonstrate the benefit of learnable parameters, we compare GBP Learning to a hand-designed denoiser: pairwise smoothing with robust factors.

We also ablate two features of our approach:

1. Continual learning vs per-frame learning
2. Single layer vs multi-layer


Video Denoising

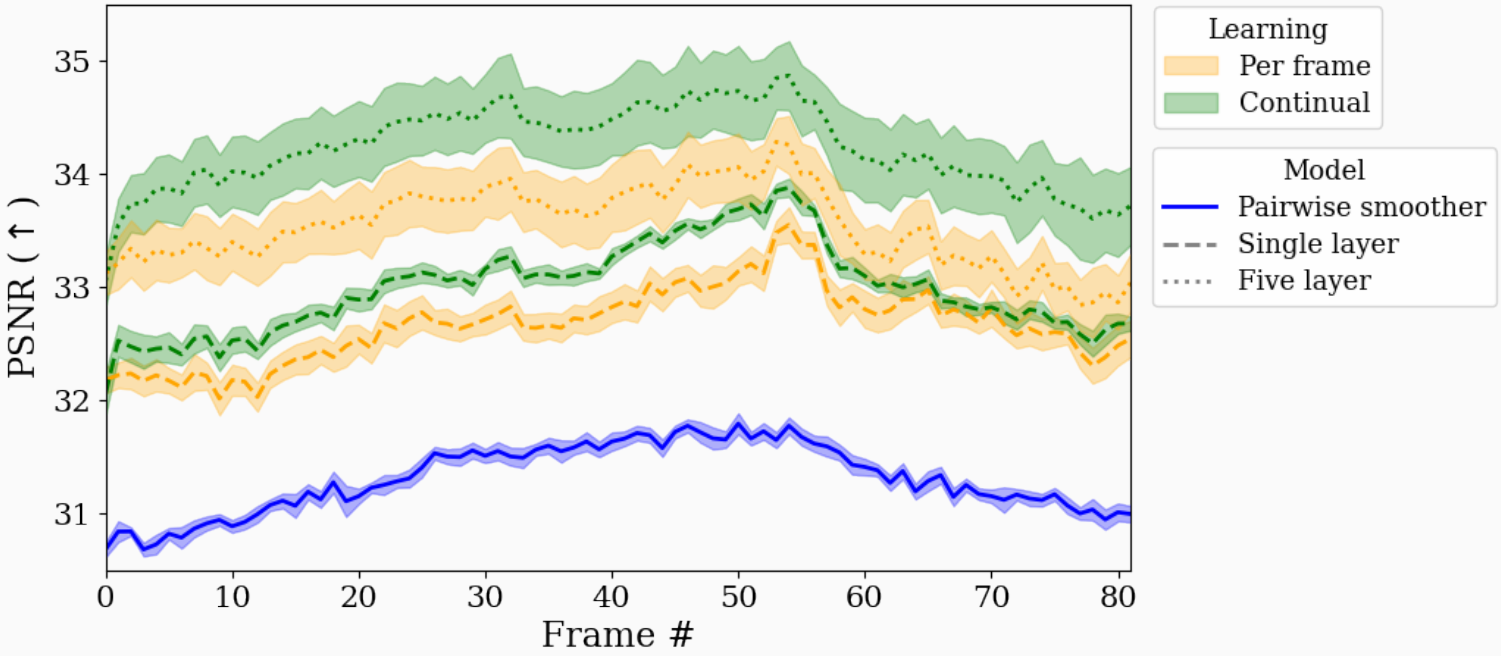
Layer #	Type	Kernel Size	# Filters	Activation
1	Transposed conv.	3×3	4	Linear

Table 2: Single layer model

Layer #	Type	Kernel Size	# Filters	Activation
1	Transposed conv.	3×3	8	Leaky ReLU
2	Upsample	2×2	N/A	N/A
3	Transposed conv.	3×3	8	Leaky ReLU
4	Upsample	2×2	N/A	N/A
5	Transposed conv.	3×3	4	Linear

Table 3: Five layer model

Video Denoising



- Continual learning outperforms learning from scratch at each frame
- Five-layer model performs better than single-layer

Video Denoising: Classical vs Learnt

(a) Clean image

(b) Corrupted image

(c) Pairwise smoothing

(d) Per-frame learning, single layer

Video Denoising: Single Layer vs Five Layer

(a) Clean image

(b) Corrupted image

(c) Per-frame learning, single layer

(d) Per-frame learning, five layer

Image Classification: MNIST

We do image classification with a small convolutional factor graph.

- The dataset is batched and the model trained via continual learning — only one epoch
- GBP is used both for training and prediction
- Parameters are fixed at test time
- Architecture as on RHS, but conv has 16 filters, kernel size 5×5

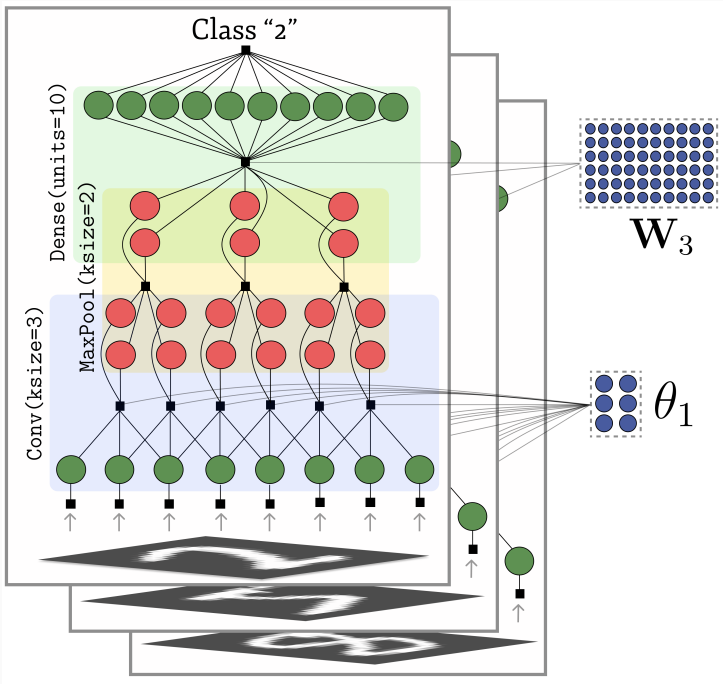
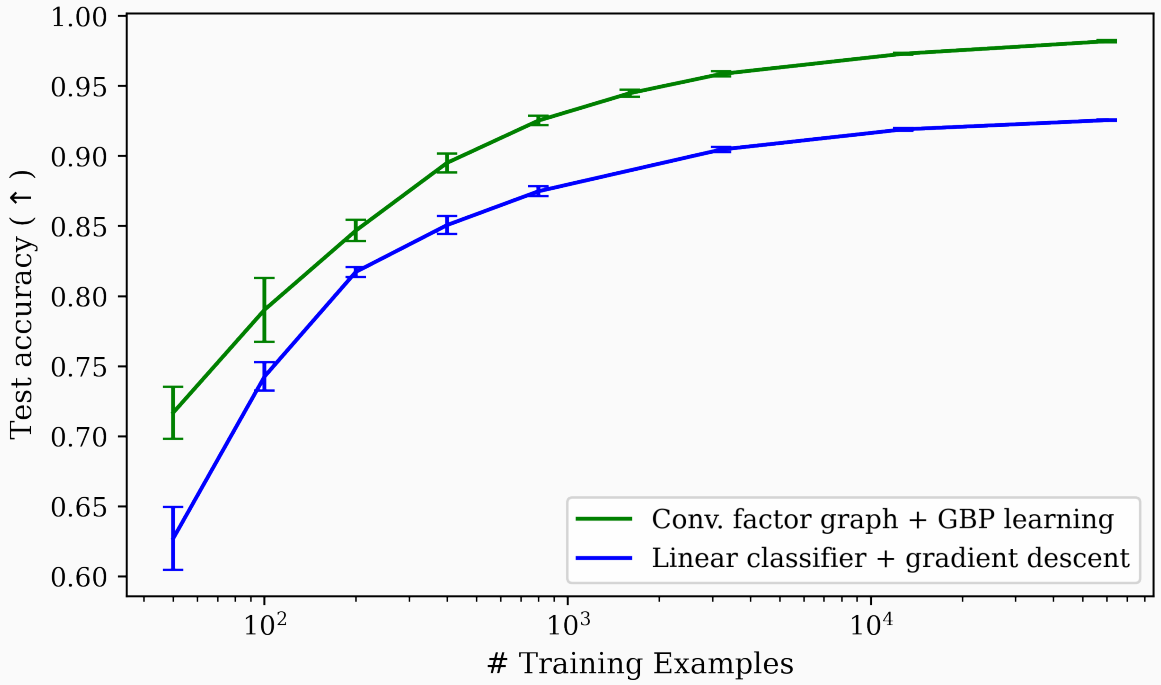


Image Classification: MNIST



- GBP Learning can effectively capture nonlinear relationships
- Also outperforms a linear classifier even in low-data regimes

Asynchronous Learning: MNIST

To simulate asynchronous training, we uniformly sample four layers to update at each iteration. Sampling is with replacement, so at each iteration only a subset of the layers are updated (and some are updated multiple times).

Asynchronous training converges in a similar number of iterations and to a similar accuracy as synchronous GBP, with forward-backward sweeps.

Asynchronous Learning: MNIST

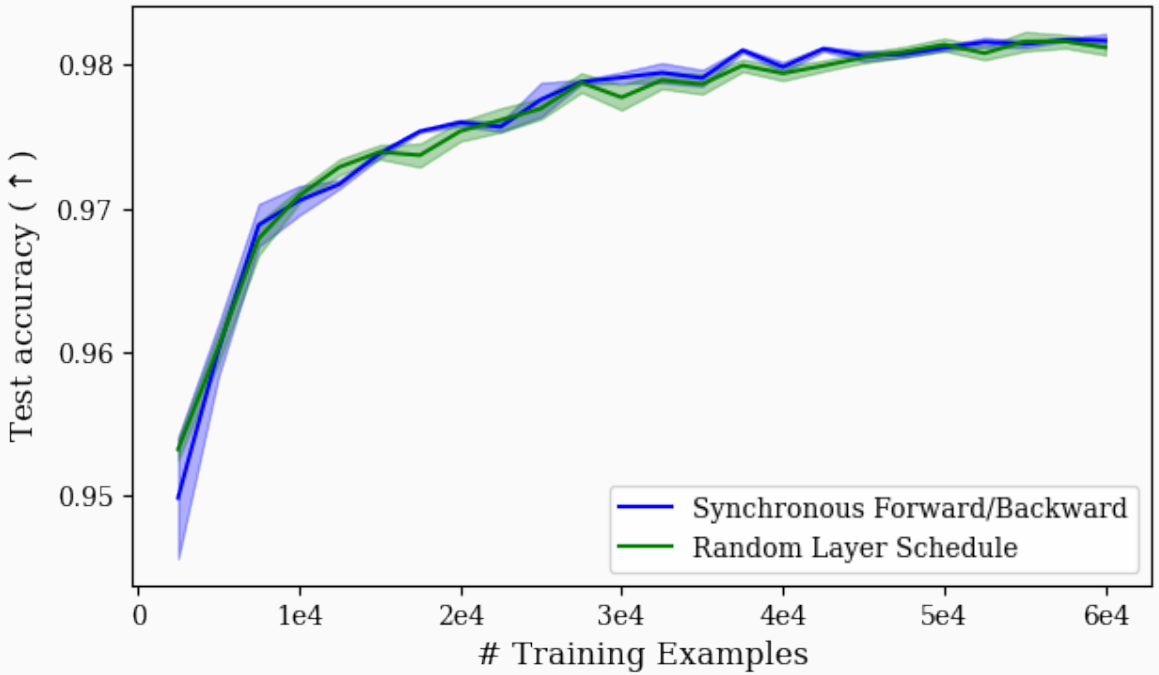


Image Classification: Comparison with Lucibello et al. (2022)

We compare the MLP-like factor graphs of Lucibello et al. (2022) with our convolutional model. We note that

- Lucibello et al. (2022) run for many ($\sim$ 100s) of epochs, we run for one,
- The derived message updates of Lucibello et al. (2022) only hold for MLP-like models, where ours is a general recipe to any architecture.

Dataset	Lucibello et al. (2022)	Ours
MNIST	97.4 ± 0.1	$\mathbf{98.2 \pm 0.1}$
FashionMNIST	88.2 ± 0.3	88.2 ± 0.3
CIFAR10	41.3 ± 0.3	$\mathbf{53.1 \pm 0.7}$

Table 4: Test accuracy (%). Ranges cover one standard deviation either side of the mean for 5 seeds.

Conclusion and Future Work

Conclusion

- We can express learnable models as Gaussian factor graphs which can be trained with GBP
- Nonlinear factors and discrete observations can be accommodated through iterative relinearisation
- Learning can be done incrementally by filtering over the parameters
- Training can be asynchronous and distributed

Future Work

- Joint state estimation and learning in a combined factor graph
- Effective scaling to deeper networks
- Graphcore IPU implementation

Thanks for listening!

Questions?

References

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Decoupled greedy learning of cnns. In *International Conference on Machine Learning*, pages 736–745. PMLR, 2020.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015.

Cerebras. Cerebras. URL https://www.cerebras.net/.

A. J. Davison and J. Ortiz. FutureMapping 2: Gaussian Belief Propagation for Spatial AI. *arXiv preprint arXiv:1910.14139*, 2019.

References ii

Frank Dellaert, Michael Kaess, et al. Factor graphs for robot perception. *Foundations and Trends® in Robotics*, 6(1-2):1–139, 2017.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

Dileep George, Wolfgang Lehrach, Ken Kansky, Miguel Lázaro-Gredilla, Christopher Laan, Bhaskara Marthi, Xinghua Lou, Zhaoshi Meng, Yi Liu, Huayan Wang, et al. A generative vision model that trains with high data efficiency and breaks text-based captchas. *Science*, 358(6368):eaag2612, 2017.

Graphcore. Graphcore. URL https://www.graphcore.ai/.

Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

References iii

Jason K Johnson, Danny Bickson, and Danny Dolev. Fixing convergence of gaussian belief propagation. In *2009 IEEE International Symposium on Information Theory*, pages 1674–1678. IEEE, 2009.

Dmitry Krotov and John J Hopfield. Unsupervised learning by competing hidden units. *Proceedings of the National Academy of Sciences*, 116(16):7723–7731, 2019.

Miguel Lázaro-Gredilla, Yi Liu, D Scott Phoenix, and Dileep George. Hierarchical compositional feature learning. *arXiv preprint arXiv:1611.02252*, 2016.

References iv

Miguel Lázaro-Gredilla, Wolfgang Lehrach, Nishad Gothoskar, Guangyao Zhou, Antoine Dedieu, and Dileep George. Query training: Learning a worse model to infer better marginals in undirected graphical models with hidden variables. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8252–8260, 2021.

Sindy Löwe, Peter O'Connor, and Bastiaan Veeling. Putting an end to end-to-end: Gradient-isolated learning of representations. *Advances in neural information processing systems*, 32, 2019.

Carlo Lucibello, Fabrizio Pittorino, Gabriele Perugini, and Riccardo Zecchina. Deep learning via message passing algorithms based on belief propagation. *Machine Learning: Science and Technology*, 3 (3):035005, 2022.

References v

David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

R. Murai, J. Ortiz, S. Saeedi, P.H.J. Kelly, and A. J. Davison. A robot web for distributed many-device localisation. *arXiv preprint arXiv:2202.03314*, 2022.

Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

J. Ortiz, M. Pupilli, S. Leutenegger, and A. J. Davison. Bundle adjustment on a graph processor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

J. Ortiz, T. Evans, and A. J. Davison. A visual introduction to Gaussian Belief Propagation. *arXiv preprint arXiv:2107.02308*, 2021.

References vi

J. Ortiz, T. Evans, E. Sucar, and A. J. Davison. Incremental Abstraction in Distributed Probabilistic SLAM Graphs. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

A. Patwardhan, R. Murai, and A. J. Davison. Distributing collaborative multi-robot planning with gaussian belief propagation. *arXiv preprint arXiv:2203.08040*, 2022.

Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. In *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings*, volume 6. International Conference on Representation Learning, 2018.

Yair Weiss and William Freeman. Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Advances in neural information processing systems*, 12, 1999.

References vii

Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. Cyclical stochastic gradient mcmc for bayesian deep learning. *arXiv preprint arXiv:1902.03932*, 2019.

GBP with Nonlinear Factors i

The factor to variable message updates for GBP are

$$\Lambda_{\phi_j \to x_k} \leftarrow \Lambda_{k,k}^{(\phi_j)} - \Lambda_{k,\backslash k}^{(\phi_j)} \left(\Lambda_{\backslash k,\backslash k}^{(\phi_j+m)}\right)^{-1} \Lambda_{\backslash k,k}^{(\phi_j)} \; ; \tag{1}$$

$$\eta_{\phi_j \to x_k} \leftarrow \eta_{k}^{(\phi_j)} - \Lambda_{k,\backslash k}^{(\phi_j)} \left(\Lambda_{\backslash k,\backslash k}^{(\phi_j+m)}\right)^{-1} \eta_{\backslash k}^{(\phi_j+m)} \; , \tag{2}$$

where $\Lambda^{(\phi_j+m)} := \Lambda^{(\phi_j)} + \mathbf{D}_{\phi_j}$ and $\eta^{(\phi_j+m)} := \eta^{(\phi_j)} + \mathbf{e}_{\phi_j}$.

For nonlinear factors we approximate the factor precision $\Lambda^{(\phi_j)}$ and information $\eta^{(\phi_j)}$ based on the local geometry. The measurement function $h(\cdot)$ relates connected variables $\mathbf{x}_j$ to an observation y_j giving the factor energy

$$E(\mathbf{x}_j, y_j) = \left(y_j - h\left(\mathbf{x}_j\right)\right)^T \Lambda_o \left(y_j - h\left(\mathbf{x}_j\right)\right). \tag{3}$$

GBP with Nonlinear Factors ii

For a Jacobian $\mathbf{J} := \frac{\partial h(\mathbf{x}_j)}{\partial \mathbf{x}_j}|_{\mathbf{x}_{j,0}}$ substituting a local linearisation of the measurement function about $\mathbf{x}_{j,0}$

$$h\left(\mathbf{x}_j\right) \approx h\left(\mathbf{x}_{j,0}\right) + \mathbf{J}\left(\mathbf{x}_j - \mathbf{x}_{j,0}\right) \tag{4}$$

into (3) we end up with an energy function quadratic in the variables. We can then compare it to the energy for a linear factor to derive the precision and information of the linearised factor (Davison and Ortiz, 2019).

$$\eta^{(\phi_j)} = \mathbf{J}^T \Lambda_o \left(\mathbf{J}\mathbf{x}_{j,0} + y_j - h\left(\mathbf{x}_{j,0}\right)\right) \tag{5}$$

$$\Lambda^{(\phi_j)} = \mathbf{J}^T \Lambda_o \mathbf{J} \tag{6}$$

Efficient Factor to Variable Messages 1 i

GBP is bottlenecked by the matrix inversion in the factor to variable message update (2). We can rewrite the inversion of $\Lambda^{(\phi_j+m)}_{\setminus k,\setminus k} := [\Lambda^{(\phi_j)} + \mathbf{D}_{\phi_j}]_{\setminus k,\setminus k}$ by first substituting the precision for that of a linearised factor (6)[1]

$$\left(\Lambda^{(\phi_j+m)}\right)^{-1} = \left(\mathbf{J}^T \Lambda_o \mathbf{J} + \mathbf{D}_{\phi_j}\right)^{-1} \tag{7}$$

and then using the Woodbury identity

$$\left(\Lambda^{(\phi_j+m)}\right)^{-1} = \mathbf{D}^{-1}_{\phi j} - \mathbf{D}^{-1}_{\phi j}\mathbf{J}^T \left(\Lambda_o^{-1} + \mathbf{J}\mathbf{D}^{-1}_{\phi j}\mathbf{J}^T\right)^{-1} \mathbf{J}\mathbf{D}^{-1}_{\phi j}. \tag{8}$$

The matrix of incoming messages $\mathbf{D}_{\phi j}$ is diagonal and so cheap to invert. Usually the observation dimension $M := \dim(y_j)$ is smaller than the number of connected variables $V_j := \dim(\mathbf{x}_j)$. Using (8), we can invert a $M \times M$ matrix instead of a $(V_j - 1) \times (V_j - 1)$ matrix. The

Efficient Factor to Variable Messages 1 ii

complexity to compute one outgoing message is reduced from $O\left((V_j - 1)^3\right)$ to $O\left((V-1)M^2 + M^3\right)$.

[1] $\backslash k$ subscripts removed for clarity

Efficient Factor to Variable Messages 2 i

For each outgoing message (2) we require intermediates which are sums over all but one of the variables, e.g.

$$\mathbf{U}_i := \mathbf{J}_{:,\setminus i} \mathbf{D}^{-1}_{\setminus i,\setminus i} \left(\mathbf{J}_{:,\setminus i}\right)^\top . \tag{9}$$

This has complexity $O\left((V-1)M^2\right)$ for *each outgoing variable* i ($\mathbf{D}^{-1}$ is diagonal). However, we can reduce this by first computing

$$\hat{\mathbf{U}} := \mathbf{J}\mathbf{D}^{-1}\mathbf{J}^\top , \tag{10}$$

which can be reused for all outgoing messages

$$\mathbf{U}_i = \hat{\mathbf{U}} - \mathbf{J}_{:,i} \mathbf{D}^{-1}_{i,i} \left(\mathbf{J}_{:,i}\right)^\top . \tag{11}$$

This reduces the complexity for computing *all* outgoing messages from ϕ_j to linear in V_j.

Video Denoising: Per-frame vs Continual Learning

(a) Clean image

(b) Corrupted image

(c) Per-frame learning, five layer

(d) Continual learning, five layer