A Other practical challenges

Parameterizing covariances It is common to to re-parameterize covariance matrices to a vector of unconstrained parameters. As above, the typical way to do this is via a function tril that maps unconstrained vectors to Cholesky factors, i.e. lower-triangular matrices with positive diagonals. This can be done by simple re-arranging the components of the vector into a lower-triangular matrix, followed by applying a function to map the entries on the diagonal components to the positive numbers. In our preliminary experiments, the choice of the mapping was quite significant in terms of how difficult optimization was. Common choices like the $\exp(x)$ and $\log(\exp(x)+1)$ functions did not perform well when the outputs were close to zero. Instead, we propose to use the transformation $x \mapsto \frac{1}{2}(x + \sqrt{x^2 + 4\gamma})$, where γ is a hyperparameter (we use $\gamma = 1$). This is based on the proximal operator for the multivariate Gaussian entropy [8, section 5]. Intuitively, when x is a large positive number, this mapping returns approximately x, while if x is a large negative number, the mapping returns approximately $-1/x$. This decays to zero more slowly than common mappings, which appears to improve numerical stability and the conditioning of the optimization.

Feature network architecture. In this paper, we propose the use of a separate `feat_net` to deal with variable-length input and order invariance. In our preliminary experiments, we found that the performance improves when we concatenate the embedding e_j in the fig. 4 with it's dimension-wise square before sending it to the pooling function `pool`. We hypothesis that this is because the embeddings act as learnable statistics, and using the elementwise square directly provides useful information to `param_net`.

Batch size selection For the small scale problems (both synthetic and MovieLens), we do not subsample data; this is to maintain a fair comparison to joint approaches that do not support sub-sampling. For moderate and large scales, we select the batch size for the branch methods based on the following rule of thumb: we increase the batch size such that $\frac{|B|}{T^{1.18}}$ is roughly maximized. This rule-of-thumb captures the following intuition. Suppose batch size $|B|$ takes time T per iteration. If the time taken per iteration for batchsize $|2B|$ is less than $1.8T$, then we should use $2B$. Of course, we roughly maximize $\frac{|B|}{T^{1.18}}$ for computational ease. We use the same batch size as the branch methods for the amortized methods as we found that amortized methods were much more robust to the choice of batchsize. We use $|B| = 200$ for moderate scale and $|B| = 400$ for large scale.

Initialization We initialize the neural network parameters using a truncated normal distribution with zero mean and standard deviation equal to $\sqrt{1/\mathrm{fan\_in}}$, where $\mathrm{fan\_in}$ is the number of inputs to the layer [24]. We initialize the final output layer of the `param-net` with a zero mean Gaussian with a standard deviation of 0.001 [1]. This ensures an almost standard normal initialization for the local conditional $q_{\texttt{net}_u(x_i,y_i)}(z_i|\theta)$.

Gradient Calculation In our preliminary experiments, we found sticking the landing gradient [30, STL] to be less stable. STL requires a re-evaluation of the density which in turn requires a matrix inversion (for the Cholesky factor); this matrix inversion was sometimes prone to numerical precision errors. Instead, we found the regular gradient, also called the total gradient in [30], to be numerically robust as it can be evaluated without a matrix inversion. This is done by simultaneously sampling and evaluating the density in much the same as done in normlaizing flows [29, 26]. We use the total gradient for all our experiments.

B General trade-offs

The amortized approach proposed in section 5 is only applicable for symmetric models. In table 3, we summarize the applicability of all the methods we discuss in this paper. In our preliminary experiments, for amortized approaches, the performance improved when we increased the number of layers in the neural networks or increased the length of the embeddings in $\texttt{net}_u$.

Table 3: Summary of method applicability.

Models	$q^{\text{branch}}_{v,w}$ (definition 1)	$q^{\text{Amort}}_{v,u}$ (definition 4)	$q^{\text{Amort}}_{v,u}$ w/ $\texttt{feat\_net}_u$ ($\texttt{net}_u$ as in fig. 4)
HBD (eq. (1))	✓	×	×
Symmetric HBD (eq. (2))	✓	✓	✓
Locally i.i.d. Symmetric HBD (eq. (3))	✓	×	✓

However, we make no serious efforts to find the optimal architecture. In fact, we use the same architecture for all our experiments, across the scales. We believe the performance on a particular task can be further improved by carefully curating the neural architecture. Note that there are no architecture choices in joint or branched approaches. We also did not optimize our choice of number of samples to drawn from q to estimate ELBO. This forms the second source of stochasticity and using more samples can help reduce the variance [34]. We use 10 copies for all our experiments.

A particularly interesting case arises when the number of local latent variables (N) is very large. In such scenarios, the true posterior $p(\theta|x, y)$ can be too concentrated. As the randomness in θ is very low, we might not gain any significant benefits from conditioning on θ—as θ reduces to a fixed quantity, Dense Gaussian will work just well as the Block Gaussian (see tables 2, 6 and 7). In practice, it is hard to know this apriori; in fact, our scalable approaches allow for such analysis on large scale model.

C Proof for Theorem

Theorem (Repeated)**.** Let p be a HBD, and $q_\phi^{\text{Joint}}(\theta, z)$ be a joint approximation family parameterized by ϕ. Choose a corresponding branch variational family $q_{v,w}^{\text{Branch}}(\theta, z)$ as in definition 1. Then,

$$\min_{v,w} KL\left(q_{v,w}^{\text{Branch}} \| p\right) \leq \min_{\phi} KL\left(q_\phi^{\text{Joint}} \| p\right).$$

Proof. Construct a new distribution q'_ϕ such that

$$q'_\phi(\theta, z) = q_\phi^{\text{Joint}}(\theta) \prod_i q_\phi^{\text{Joint}}(z_i|\theta). \tag{18}$$

Then, note that z_i are conditionally independent in q'_ϕ, such that,

$$q'_\phi(z_i|\theta, z_{<i}) = q'_\phi(z_i|\theta). \tag{19}$$

From chain rule of KL-divergence, we have

$$\begin{aligned}
KL\left(q_\phi^{\text{Joint}}(\theta, \mathsf{z}) \| p(\theta, \mathsf{z}|x, y)\right) &= KL\left(q_\phi^{\text{Joint}}(\theta) \| p(\theta|x, y)\right) \\
&\quad + \sum_i KL\left(q_\phi^{\text{Joint}}(\mathsf{z}_i|\mathsf{z}_{<i}, \theta) \| p(\mathsf{z}_i|\mathsf{z}_{<i}, \theta, x, y)\right), \text{ and} \\
KL\left(q'_\phi(\theta, \mathsf{z}) \middle\| p(\theta, \mathsf{z}|x, y)\right) &= KL\left(q_\phi^{\text{Joint}}(\theta) \| p(\theta|x, y)\right) \\
&\quad + \sum_i KL\left(q'_\phi(\mathsf{z}_i|\mathsf{z}_{<i}, \theta) \middle\| p(\mathsf{z}_i|\mathsf{z}_{<i}, \theta, x, y)\right).
\end{aligned}$$

Consider any arbitrary summand term. We have that

$$\begin{aligned}
&KL\left(q_\phi^{\text{Joint}}(\mathsf{z}_i|\mathsf{z}_{<i}, \theta) \| p(\mathsf{z}_i|\mathsf{z}_{<i}, \theta, x, y)\right) \\
&\overset{(1)}{=} KL\left(q_\phi^{\text{Joint}}(\mathsf{z}_i|\mathsf{z}_{<i}, \theta) \| p(\mathsf{z}_i|\theta, x_i, y_i)\right) \\
&\overset{(2)}{=} \mathop{\mathbb{E}}_{\theta \sim q_\phi^{\text{Joint}}(\theta)} \mathop{\mathbb{E}}_{z_{<i} \sim q_\phi^{\text{Joint}}(\mathsf{z}_{<i}|\theta)} \left[KL\left(q_\phi^{\text{Joint}}(\mathsf{z}_i|z_{<i}, \theta) \| p(\mathsf{z}_i|\theta, x_i, y_i)\right)\right] \\
&\overset{(3)}{\geq} \mathop{\mathbb{E}}_{\theta \sim q_\phi^{\text{Joint}}(\theta)} \left[KL\left(\mathop{\mathbb{E}}_{z_{<i} \sim q_\phi^{\text{Joint}}(\mathsf{z}_{<i}|\theta)} \left[q_\phi^{\text{Joint}}(\mathsf{z}_i|z_{<i}, \theta)\right] \middle\| \mathop{\mathbb{E}}_{z_{<i} \sim q_\phi^{\text{Joint}}(\mathsf{z}_{<i}|\theta)} \left[p(\mathsf{z}_i|\theta, x_i, y_i)\right]\right)\right] \\
&\overset{(4)}{=} \mathop{\mathbb{E}}_{\theta \sim q_\phi^{\text{Joint}}(\theta)} \left[KL\left(q_\phi^{\text{Joint}}(\mathsf{z}_i|\theta) \| p(\mathsf{z}_i|\theta, x_i, y_i)\right)\right] \\
&= KL\left(q_\phi^{\text{Joint}}(\mathsf{z}_i|\theta) \| p(\mathsf{z}_i|\theta, x_i, y_i)\right) \\
&= KL\left(q'_\phi(\mathsf{z}_i|\theta) \middle\| p(\mathsf{z}_i|\theta, x_i, y_i)\right) \\
&\overset{(5)}{=} KL\left(q'_\phi(\mathsf{z}_i|\theta, \mathsf{z}_{<i}) \middle\| p(\mathsf{z}_i|\theta, \mathsf{z}_{<i}, x_i, y_i)\right),
\end{aligned}$$

where (1) follows from HBD structure; (2) follows from definition of conditional KL divergence; (3) follows from convexity of KL divergence and Jensen's inequality; (4) follows from marginalization, and (5) follows from the conditional independence of q' and p. Summing the above result over i gives that

$$KL\left(q'_\phi \| p\right) \leq KL\left(q_\phi^{\text{Joint}} \| p\right). \tag{20}$$

Now, from definition 1, we know that for every ϕ, there exists a corresponding (v, w) such that $q_{v,w}^{\text{Branch}} = q'_\phi$. Let $\phi^* = \text{argmin}_\phi \, KL\left(q_\phi^{\text{Joint}} \| p\right)$. Then, there exists some $q_{v,w}^{\text{Branch}} = q'_{\phi^*}$. Then, it follows that

$$\min_{v,w} KL\left(q_{v,w}^{\text{Branch}} \| p\right) \leq KL\left(q'_{\phi^*} \| p\right) \leq KL\left(q_{\phi^*}^{\text{Joint}} \| p\right) = \min_{\phi} KL\left(q_\phi^{\text{Joint}} \| p\right). \tag{21}$$

□

D Proof for Claim

Claim (Repeated)**.** Let p be a symmetric HBD and let q_ϕ^{Joint} be some joint approximation. Let $q_{v,u}^{\text{Amort}}$ be as in definition 1. Suppose that for all v, there exists a u, such that,

$$\texttt{net}_u(x_i, y_i) = \underset{w_i}{\text{argmax}} \mathop{\mathbb{E}}_{q_v(\theta)} \mathop{\mathbb{E}}_{q_{w_i}(\mathsf{z}_i|\theta)} \left[\log \frac{p\left(\mathsf{z}_i, y_i|\theta, x_i\right)}{q_{w_i}\left(\mathsf{z}_i|\theta\right)}\right]. \tag{22}$$

Then,

$$\min_{v,u} KL\left(q_{v,u}^{\text{Amort}} \| p\right) \leq \min_{\phi} KL\left(q_\phi^{\text{Joint}} \| p\right) \tag{23}$$

Proof. Consider the optimization for $q_{v,w}^{\text{Branch}}$. We have

$$\begin{aligned}
\max_{v,w} \mathcal{L}\left(q_{v,w}^{\text{Branch}} \| p\right) &= \max_{v,w} \left[\mathop{\mathbb{E}}_{q_v(\theta)} \left[\log \frac{p\left(\theta\right)}{q_v\left(\theta\right)}\right] + \sum_{i=1}^{N} \mathop{\mathbb{E}}_{q_v(\theta)} \mathop{\mathbb{E}}_{q_{w_i}(\mathsf{z}_i|\theta)} \left[\log \frac{p\left(\mathsf{z}_i, y_i|\theta, x_i\right)}{q_{w_i}\left(\mathsf{z}_i|\theta\right)}\right]\right] \\
&= \max_{v} \left[\mathop{\mathbb{E}}_{q_v(\theta)} \left[\log \frac{p\left(\theta\right)}{q_v\left(\theta\right)}\right] + \sum_{i=1}^{N} \max_{w_i} \mathop{\mathbb{E}}_{q_v(\theta)} \mathop{\mathbb{E}}_{q_{w_i}(\mathsf{z}_i|\theta)} \left[\log \frac{p\left(\mathsf{z}_i, y_i|\theta, x_i\right)}{q_{w_i}\left(\mathsf{z}_i|\theta\right)}\right]\right] \\
&\overset{(1)}{=} \max_{v} \left[\mathop{\mathbb{E}}_{q_v(\theta)} \left[\log \frac{p\left(\theta\right)}{q_v\left(\theta\right)}\right] + \sum_{i=1}^{N} \mathop{\mathbb{E}}_{q_v(\theta)} \mathop{\mathbb{E}}_{q_{\texttt{net}_u(x_i,y_i)}(\mathsf{z}_i|\theta)} \left[\log \frac{p\left(\mathsf{z}_i, y_i|\theta, x_i\right)}{q_{\texttt{net}_u(x_i,y_i)}\left(\mathsf{z}_i|\theta\right)}\right]\right] \\
&\leq \max_{u} \max_{v} \left[\mathop{\mathbb{E}}_{q_v(\theta)} \left[\log \frac{p\left(\theta\right)}{q_v\left(\theta\right)}\right] + \sum_{i=1}^{N} \mathop{\mathbb{E}}_{q_v(\theta)} \mathop{\mathbb{E}}_{q_{\texttt{net}_u(x_i,y_i)}(\mathsf{z}_i|\theta)} \left[\log \frac{p\left(\mathsf{z}_i, y_i|\theta, x_i\right)}{q_{\texttt{net}_u(x_i,y_i)}\left(\mathsf{z}_i|\theta\right)}\right]\right] \\
&= \max_{u} \max_{v} \mathcal{L}\left(q_{v,u}^{\text{Amort}} \| p\right),
\end{aligned}$$

where (1) follows from the assumption in the Claim. Now, from the ELBO decomposition equation, we have

$$\log p(y|x) = \mathcal{L}\left(q \| p\right) + KL\left(q \| p\right). \tag{24}$$

Therefore, we have

$$\min_{u} \min_{v} KL\left(q_{v,u}^{\text{Amort}} \| p\right) \leq \min_{v,w} KL\left(q_{v,w}^{\text{Branch}} \| p\right) \tag{25}$$

From theorem 2, we get the desired result.

$$\min_{u} \min_{v} KL\left(q_{v,u}^{\text{Amort}} \| p\right) \leq \min_{v,w} KL\left(q_{v,w}^{\text{Branch}} \| p\right) \leq \min_{\phi} KL\left(q_\phi^{\text{Joint}} \| p\right) \tag{26}$$

□

E Derivation for Branch Gaussian

Let $q_\phi^{\text{joint}}(\theta, z) = \mathcal{N}((\theta, z)|\mu, \Sigma)$ be the joint Gaussian approximation as in corrolary 3. Further, let (μ, Σ) be defined as

$$\mu = \begin{bmatrix} \mu_\theta \\ \mu_{z_1} \\ \vdots \\ \mu_{z_N} \end{bmatrix} \quad \text{and} \quad \Sigma = \begin{bmatrix} \Sigma_\theta & \Sigma_{\theta z_1} & \dots & \Sigma_{\theta z_N} \\ \Sigma_{\theta z_1}^\top & \Sigma_{z_1} & \dots & \Sigma_{z_1 z_N} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{\theta z_n}^\top & \Sigma_{z_1 z_N}^\top & \dots & \Sigma_{z_N} \end{bmatrix}. \tag{27}$$

Then, from the properties of the multivariate Gaussian [27]

$$q_\phi^{\text{joint}}(z_i|\theta) = \mathcal{N}(z_i|\mu_{z_i|\theta}, \Sigma_{z_i|\theta}), \text{ where} \tag{28}$$

$$\mu_{z_i|\theta} = \mu_{z_i} + \Sigma_{\theta z_i}^\top \Sigma_\theta^{-1}(\theta - \mu_\theta), \text{ and} \tag{29}$$

$$\Sigma_{z_i|\theta} = \Sigma_{z_i z_i} - \Sigma_{\theta z_i}^\top \Sigma_\theta^{-1} \Sigma_{\theta z_i}. \tag{30}$$

Now, to parameterize a corresponding $q_{v,w}^{\text{Branch}}$, we use the (μ_i, Σ_i, A_i), such that,

$$q_{v,w}^{\text{Branch}}(z_i|\theta) = \mathcal{N}(z_i|\mu_i + A_i\theta, \Sigma_i). \tag{31}$$

F Experimental Details

Architectural Details We use the architecture as reported in table 4 for all our amortized approaches. In addition to using e_j as detailed in fig. 4, we concatenate the elementwise square before sending it to `param-net`. Thus, the input to `param-net` is not 128 dimensional but 256 dimensional. Further, we use mean as the `pool` function.

Table 4: Architecture details for $\texttt{net}_u$. Each fully-connected layer is followed by leaky-ReLU baring the last layer.

Network	Layer Skeleton
`feat-net`	64, 64, 64, 128
`param-net`	256, 256, 256

Compute Resources We use JAX [6] to implement our methods. We trained using Nvidia 2080ti-12GB. All methods finished training within 4 hours. Branch approaches were at an average twice as fast as amortized variants.

Step-size drop We use Adam [18] for training with an initial step-size of 0.001 (and default values for other hyperparameters.) In preliminary experiments, we found that dropping the step-size improves the performance. Starting from 0.001, we drop the step to one-tenth of it's value after a predetermined number of steps. For small scale experiments, we drop a total of three times after every 50,000 iterations (we train for 200,000 iterations.) For moderate and large scale models, we drop once after 100,000 iterations.

F.1 Movielens

Feature Dimensions We reduce the movie feature dimensionality to 10 using PCA. This is done with branch approaches in focus as the number of features for dense branch Gaussian scale as $\mathcal{O}(ND^3)$, where D is the dimensionality of the movie features. Note, that the number of features for amortized approaches is independent of N allowing for better scalability.

Table 5: Metrics used for evaluation. We use K = 10,000 samples from the posterior. Here, $(z^k, \theta^k) \sim q(\mathbf{z}, \theta|x^{\text{train}}, y^{\text{train}})$.

Metric	Expression
Test likelihood	$\log \frac{1}{K} \sum_k p(y^{\text{test}}|x^{\text{test}}, z^k, \theta^k)$
Train likelihood	$\log \frac{1}{K} \sum_k \frac{p(y^{\text{train}}, z^k, \theta^k|x^{\text{train}})}{q(z^k, \theta^k|x^{\text{train}}, y^{\text{train}})}$
Train ELBO	$\frac{1}{K} \sum_k \log \frac{p(y^{\text{train}}, z^k, \theta^k|x^{\text{train}})}{q(z^k, \theta^k|x^{\text{train}}, y^{\text{train}})}$

Metrics We use three metrics for performance evaluation—test likelihood, train likelihood, and train ELBO. Details of the expressions are presented in table 5. We draw a batch of fresh 10,000 samples from the posterior to estimate each metric. Of course, the evaluated expressions are just approximation to the true value. In table 6 and table 7 we present the extended results. In table 7 we present the same values but normalized by the number of ratings in the dataset.

Table 6: This table has the extended results for the Movielens25M Dataset. All values are in nats. Higher is better.

		Test LL			Train LL			Train ELBO		
≈ # ratings		2.5K	180K	18M	2.5K	180K	18M	2.5K	180K	18M
Methods										
Dense	q_ϕ^{Joint}	-166.37			-1373.97			-1572.31		
	$q_{v,w}^{\text{Branch}}$	-166.66	-11054.43	-1.3046e+06	-1374.20	-95731.42	-1.0315e+07	-1572.39	-1.0368e+05	-1.1413e+07
	$q_{v,u}^{\text{Amort}}$	-166.64	-10976.38	-1.1476e+06	-1374.27	-95980.37	-1.0027e+07	-1572.45	-1.0352e+05	-1.0665e+07
Block Diagonal	q_ϕ^{Joint}	-167.36			-1375.56			-1579.04		
	$q_{v,w}^{\text{Branch}}$	-166.97	-10987.17	-1.2538e+06	-1375.71	-95891.42	-1.0399e+07	-1579.05	-1.0350e+05	-1.1078e+07
	$q_{v,u}^{\text{Amort}}$	-166.96	-10975.96	-1.1484e+06	-1375.71	-95962.56	-1.0027e+07	-1579.06	-1.0353e+05	-1.0665e+07
Diagonal	q_ϕ^{Joint}	-167.39			-1377.25			-1592.59		
	$q_{v,w}^{\text{Branch}}$	-167.31	-10977.95	-1.2713e+06	-1377.19	-96414.40	-1.0709e+07	-1592.64	-1.0428e+05	-1.1325e+07
	$q_{v,u}^{\text{Amort}}$	-167.29	-10980.75	-1.1497e+06	-1377.20	-96467.88	-1.0068e+07	-1592.64	-1.0430e+05	-1.0736e+07

Table 7: This table has the extended results for the Movilens25M Dataset. It has the same results as in table 6; however, the values are divided by the number of ratings.

		Test LL			Train LL			Train ELBO		
	≈ # ratings	2.5K	180K	18M	2.5K	180K	18M	2.5K	180K	18M
Methods										
Dense	q_ϕ^{Joint}	-0.5717			-0.5108			-0.5845		
	$q_{v,w}^{\text{Branch}}$	-0.5727	-0.5640	-0.6486	-0.5109	-0.5224	-0.5492	-0.5845	-0.5658	-0.6077
	$q_{v,u}^{\text{Amort}}$	-0.5726	-0.5600	-0.5705	-0.5109	-0.5238	-0.5339	-0.5846	-0.5649	-0.5678
Block Diagonal	q_ϕ^{Joint}	-0.5751			-0.5114			-0.5870		
	$q_{v,w}^{\text{Branch}}$	-0.5738	-0.5606	-0.6233	-0.5114	-0.5233	-0.5537	-0.5870	-0.5648	-0.5898
	$q_{v,u}^{\text{Amort}}$	-0.5738	-0.5600	-0.5709	-0.5114	-0.5237	-0.5339	-0.5870	-0.5650	-0.5678
Diagonal	q_ϕ^{Joint}	-0.5752			-0.5120			-0.5920		
	$q_{v,w}^{\text{Branch}}$	-0.5749	-0.5601	-0.6320	-0.5120	-0.5261	-0.5702	-0.5921	-0.5691	-0.6029
	$q_{v,u}^{\text{Amort}}$	-0.5749	-0.5602	-0.5716	-0.5120	-0.5264	-0.5360	-0.5921	-0.5691	-0.5716

Preprocess Movielens25M originally uses a 5 point ratings system. To get binary ratings, we map ratings greater than 3 points to 1 and less than and equal to 3 to 0.

F.2 Synthetic problem

Details of the model We use the hierarchical regression model

$$p(\theta, z, y|x) = \mathcal{N}(\theta|0, I) \prod_{i=1}^{N} \mathcal{N}(z_i|\theta, I) \prod_{j=1}^{n_i} \mathcal{N}(y_{ij}|x_{ij}^\top z_i, 1)$$

for synthetic experiments. For simplicity, we use $n_i = 100$ for all i; we vary N to create different scale variants—we use $N = 10$ for small scale, $N = 1000$ for moderate scale, and $N = 100000$ for large scale experiments; we set $x_{ij} \in \mathbb{R}^{10}$ and thus $\theta \in \mathbb{R}^{10}$ and $z_i \in \mathbb{R}^{10}$; $y_{ij} \in \mathbb{R}$.

Expression for posterior

$$p(\theta|x, y) = \mathcal{N}\Bigg(\left[I_D + \sum_i x_i^\top (I_{M_i} + x_i x_i^\top)^{-1} x_i \right]^{-1} \left[\sum_i x_i^\top (I_M + x_i x_i^\top)^{(-1)} y_i \right], \left[I_D + \sum_i x_i^\top (I_M + x_i x_i^\top)^{-1} x_i \right]^{-1} \Bigg)$$

$$p(z_i|\theta, x_i, y_i) = \mathcal{N}([I_D + x_i^\top x_i]^{-1}[x_i^\top y_i + \theta], [I_D + x_i^\top x_i]^{-1})$$

Expression for marginal likelihood

$$p(y|x) = \mathcal{N}\left(0, \begin{bmatrix} I_M + 2x_1 x_1^\top & \dots & x_n x_1^\top \\ \vdots & \ddots & \vdots \\ x_n x_1^\top & \dots & I_M + 2x_n x_n^\top \end{bmatrix}\right)$$

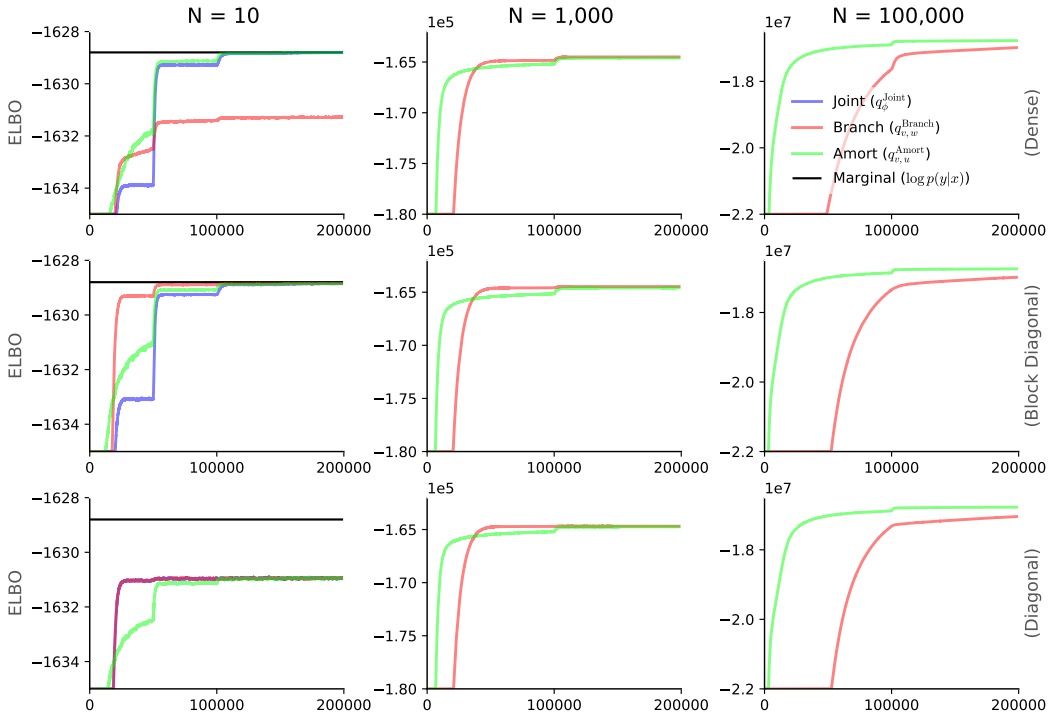

Figure 7: Training ELBO trace for the synthetic problem. Top to bottom: dense, block diagonal, and diagonal Gaussian (for each, we have q_ϕ^{Joint}, $q_{v,w}^{\text{Branch}}$, and $q_{v,u}^{\text{Amort}}$ method.) Left to right: small, moderate, and large scale of the synthetic problem. For clarity, we plot the exponential moving average of the training ELBO trace with a smoothing value of 0.001. For the small setting, we also plot the true log-marginal $\log p(y|x)$ for reference (black horizontal line): ELBO for dense approach is exactly same as the log-marginal, it's slightly lower for block, and is much less for the diagonal (see first column.) Note, calculating the log-marginal was computationally prohibitive for the moderate and large setting.