Consider a recognition-parametrized model

$$p_\theta(\mathcal{X}, \mathcal{Z}) = p_\theta(\mathcal{Z}) \prod_j \frac{f_{\theta_j}(\mathcal{Z}|\mathbf{x}_j) p_j(\mathbf{x}_j)}{F_{\theta_j}(\mathcal{Z})}$$

with exponential family $f_{\theta_j}(\mathcal{Z}|\mathbf{x}_j), p(\theta(\mathcal{Z})$ trained via (a lower bound to) variational free energy

$$\begin{aligned}
\log p_\theta(\mathcal{X}) &\geq \mathbb{E}_{q(\mathcal{Z}|\mathcal{X})}\left[\log \frac{p_\theta(\mathcal{X}, \mathcal{Z})}{q_\psi(\mathcal{Z}|\mathcal{X})}\right] \\
&= \mathbb{E}_q[\log p_\theta(\mathcal{Z})] + \sum_j \mathbb{E}_q[\log f_{\theta_j}(\mathcal{Z}|\mathbf{x}_j)] + H[q] - \sum_j \mathbb{E}_q[\log F_{\theta_j}(\mathcal{Z})] + const. \\
&\geq \mathbb{E}_q[\log p_\theta(\mathcal{Z})] + \sum_j \mathbb{E}_q[\log f_{\theta_j}(\mathcal{Z}|\mathbf{x}_j)] - (1+J)H[q] - \sum_j \log \int F_{\theta_j}(\mathcal{Z})/h_j(\mathcal{Z}) d\mathcal{Z} - \mathbb{E}_q[\log h_j(\mathcal{Z})] + const.
\end{aligned}$$

where $h_j(\mathcal{Z}) = \exp(\tilde{\eta}_j(\mathcal{X})^\top t(\mathcal{Z}))$ with inner variational parameters $\tilde{\eta}_j$ (one per data-point $\mathcal{X}$).

There are several ways we can handle the recognition model $q(\mathcal{Z}|\mathcal{X})$, even if for all of them we assume $q(\mathcal{Z}|\mathcal{X}) = q(\mathcal{Z}|\eta_q(\mathcal{X}))$ to lie in the same exponential family as $f_{\theta_j}(\mathcal{Z}|\mathbf{x}_j), p_\theta(\mathcal{Z})$:

- $\eta_q(\mathcal{X}^{(n)}) = \eta_q^{(n)}$: nonparametric natural parameter model (VI) with own natural parameter per datum n.
- $\eta_q(\mathcal{X}^{(n)}) = NN_\psi(\mathcal{X}^{(n)})$: parametric natural parameter model (VAE) with recognition parameters ψ.
- $\eta_q(\mathcal{X}^{(n)}) = \sum_j \eta_{\theta_j}(\mathbf{x}_j^{(n)}) + (1-J)\eta_0$: analytic approach, holds if we assume $\forall j : F_{\theta_j} = p_\theta(\mathcal{Z})$.

Similarly, we can handle inner variational parameters $\tilde{\eta}_j(\mathcal{X})$ as

- $\forall j : \tilde{\eta}_j(\mathcal{X}^{(n)}) = \tilde{\eta}_j^{(n)}$: nonparametric.
- $\forall j : \tilde{\eta}_j(\mathcal{X}^{(n)}) = NN_{\psi_j}(\mathcal{X}^{(n)})$: parametric with parameters $\{\psi_j\}_j$.
- $\forall j : \tilde{\eta}_j(\mathcal{X}^{(n)}) = \eta_0 - \eta_q(\mathcal{X}^{(n)})$: Hugo's Ansatz with whatever choice for $\eta_q(\mathcal{X})$ we took above.

For $\mathcal{Z} = [\mathcal{Z}_1, \ldots, \mathcal{Z}_t, \ldots, \mathcal{Z}_T]$, a state-space version of the RPM is given by

$$p_\theta(\mathcal{X}, \mathcal{Z}) = p_\theta(\mathcal{Z}) \prod_j \prod_t \frac{f_{\theta_j}(\mathcal{Z}_t|\mathbf{x}_{jt}) p_{jt}(\mathbf{x}_{jt})}{F_{\theta_j}(\mathcal{Z}_t)}.$$

Since a time-series prior will introduce temporal correlations among the $\mathcal{Z}_t$, we **do no** expect $\forall j : \prod_t F_{\theta_j}(\mathcal{Z}_t) \approx p_\theta(\mathcal{Z})$ in case of a good fit. Instead we'd like

$$\forall j, t : F_{\theta_j}(\mathcal{Z}_t) \approx p_\theta(\mathcal{Z}_t)$$

with prior marginals $p_\theta(\mathcal{Z}_t)$. For an amortized posterior $q_\psi(\mathcal{Z}|\mathcal{X})$ within the exponential family, based on the generative recognition-parametrized model $p_\theta(\mathcal{X}, \mathcal{Z})$, the corresponding form is thus

$$q_\theta(\mathcal{Z}|\mathcal{X}) = \frac{\tilde{p}_\psi(\mathcal{Z}) \prod_j \prod_t f_{\theta_j}(\mathcal{Z}_t|\mathbf{x}_{jt})}{\int \tilde{p}_\psi(\mathcal{Z}') \prod_j \prod_t f_{\theta_j}(\mathcal{Z}'_t|\mathbf{x}_{jt}) d\mathcal{Z}'}$$

with $\tilde{p}_\psi(\mathcal{Z}) = p_\theta(\mathcal{Z}) \ / \ \left(\prod_t p_\theta(\mathcal{Z}_t)\right)^J$.

Assume that $p_\theta(\mathcal{Z}) \in \text{ExpFam}[t, \chi]$ with natural parameter η_0 and that the marginals $p_\theta(\mathcal{Z}_t)$ are such that $\prod_t p_\theta(\mathcal{Z}_t) \in \text{ExpFam}[t, \chi]$ with natural parameter η_0^{unc}. Then if also $\prod_t f_{\theta_j}(\mathcal{Z}_t|\mathbf{x}_{jt}) \in \text{ExpFam}[t, \chi]$, we have $q_\theta(\mathcal{Z}|\mathcal{X}) \in \text{ExpFam}[t, \chi]$ with

$$\eta_q(\mathcal{X}) = \eta_0 - J\eta_0^{unc} + \sum_j \eta_{\theta_j}^{unc}(\mathbf{x}_j).$$

1 Experiments

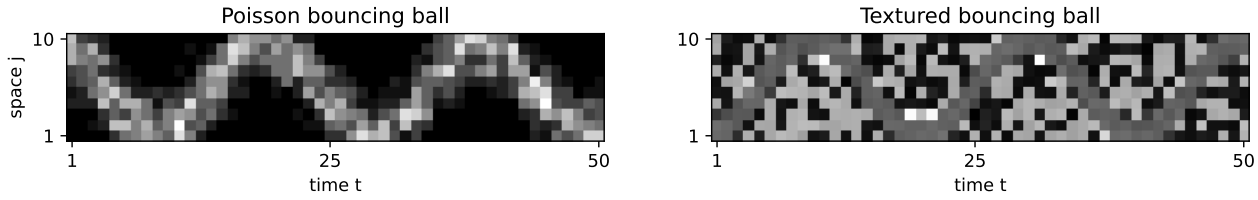

Figure 1: Example single observations $\mathcal{X}$ (dimension $J \times T = 10 \times 50$) for the two bouncing ball experiments.

We reproduce the bouncing ball experiments from Walker et al. (2023), i.e. the Poisson ball and the textured ball experiments (Fig 1), with dataset sizes $N = 10$ and $N = 100$. We use $J = 10$ (where according to personal conversation, Walker et al. used $J = 1$ for the model, ignoring conditional independence across rows of $\mathcal{X}$).

2 RPM training frameworks

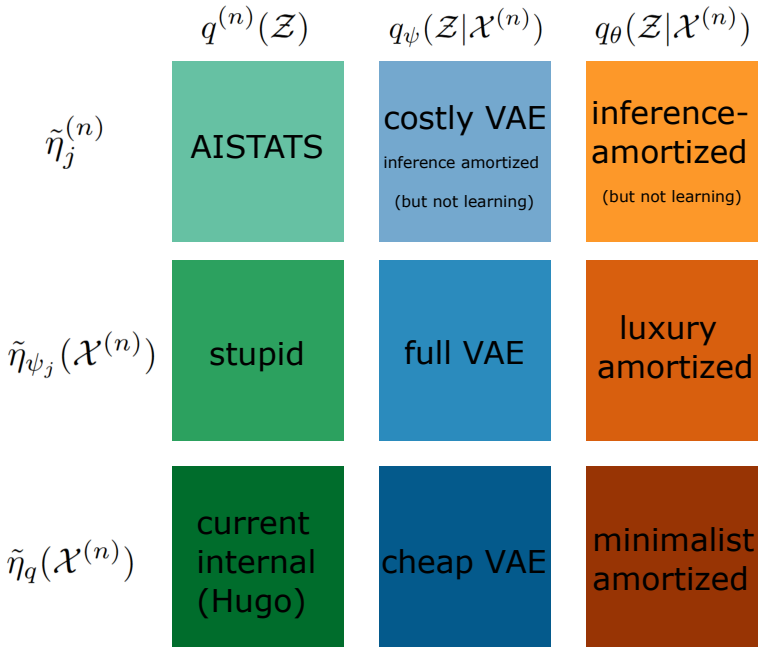

Figure 2: All tested combinations of parametrizing the recognition model $q(\mathcal{Z}|\mathcal{X})$ and the inner variational parameters $\tilde{\eta}_j(\mathcal{X}), j = 1, \ldots, J$. The green column includes training frameworks that have been tested before, the orange column is the suggestion mainly tested here. The VAEs (blue column) are mostly meant as baselines.

We compare all training frameworks in figure 2 and several (basic) initialization methods for parameter collections $(\theta, \psi, \{\psi_j\})$ of generative and recognition models:

- $(\theta, \psi, \{\psi_j\})$ completely random, only ensuring valid natural parameters, i.e. positive variances etc.
- pre-training $\eta_q(\mathcal{X}, \psi), \tilde{\eta}_j(\mathbf{x}_j, \psi_j)$, where appropriate, to close the variational gap via optimizing the ELBO (keeping θ fixed).
- pre-training $\eta_q(\mathcal{X}, \psi), \tilde{\eta}_j(\mathbf{x}_j, \psi_j)$, where appropriate, to match $\eta_q(\mathcal{X}) = \eta_0 - J\eta_0^{unc} + \sum_j \eta_{\theta_j}^{unc}(\mathbf{x}_j)$ and $\tilde{\eta}_j(\mathcal{X}) = \eta_0 - \eta_q(\mathcal{X})$, via a mean-square loss (should really be the Bregman divergence, but MSE worked).

We are mostly interested in how different ways of handling the (inner) recognition models compare – the point of the pre-trained initializations is to reduce variability of the initializaitons in $\psi, \{\psi_j\}$ (but not in θ), so we don't have to run as many random seeds.

Lastly, we test both the time-series RPM formulation with $p_\theta(\mathbf{x}_j|\mathcal{Z}) = \prod_t f_{\theta_j}(\mathcal{Z}_t|\mathbf{x}_{jt}) p_{jt}(\mathbf{x}_j t)/F_{\theta_j}(\mathcal{Z}_t)$ and the standard formulation $p_\theta(\mathbf{x}_j|\mathcal{Z}) = f_{\theta_j}(\mathcal{Z}|\mathbf{x}_j) p_j(\mathbf{x}_j)/F_{\theta_j}(\mathcal{Z})$ that ignores conditional independence across time (standard RPM results in appendix A.7).

Since the total parameter counts between RPM training frameworks differ drastically (Fig A.1 and Fig A.2), we also include a variant of the 'minimalist amortized' RPM training framework that matches the parameter count of the 'full VAE' framework. This parameter-matched variant of 'minimalist amortized' has recognition factors $f_{\theta_j}(\mathcal{Z}|\mathbf{x}_j)$ with 33 hidden units per layer in the underlying neural networks instead of the 20 hidden units in all other models in this study.

3 Results

3.1 Model initialization checks

For starters, we compare model initializations (i: random, ii: pre-train $q_\psi, \{\tilde{\eta}_j\}$ using ELBO, iii: pre-train $q_\psi, \{\tilde{\eta}_j\}$ using $F_{\theta_j}(\mathcal{Z}) = p_\theta(\mathcal{Z})$ and $\tilde{\eta}_j = \eta_0 - \eta_q$ assumptions)) for $N = 10$ and the time-series RPM variant across all $3 \times 3 = 9$ RPM training frameworks depicted in Fig 2.

In summary (Figs 3-5), the three different methods of initialization yield comparable results, meaning that for larger fits ($N = 100$, next section) we will just focus on one of them.

Overall, the 'minimalist amortized' (brown color) RPM framework **works decently on the Poisson bouncing balls**, converging much quicker than any of the non-amortized training frameworks, but performing worse than the VAE frameworks (i.e. those with separate neural network for the recognition model $q(\mathcal{Z}|\mathcal{X})$), even when controlling for the different number of free parameters (33-hidden-unit variant, dashed brown). On the **textured** bouncing balls, the 'minimalist amortized' framework works flat out best with either 20 or 33 hidden units, potentially due to simple overfitting of the VAE frameworks (Fig A.4 and A.7) and general convergence issues with non-parametric (VI) frameworks. We try to address the VI frameworks' convergence issues in the next section with overall more training epochs.

From these two experiments, it also seems best to apply the $F_{\theta_j}(\mathcal{Z}) = p_\theta(\mathcal{Z})$ assumption to either both η_q and $\tilde{\eta}_j = \eta_0 - \eta_q$, or not at all: In either experiments, combining the analytic posterior $\eta_q = \eta_0 - J\eta_0^{\text{unc}} + \sum_j \eta_j^{unc}$ with non-parametric inner-VI parameters $\tilde{\eta}_j^{(n)}$ ('inference-amortized', bright orange) yields catastrophic results, and on the textured bouncing balls also the combination with separate inner recognition models for $\tilde{\eta}_j(\mathcal{X}) = NN_{\psi_j}(\mathcal{X})$ ('luxury amortized', dark orange) worked terribly.

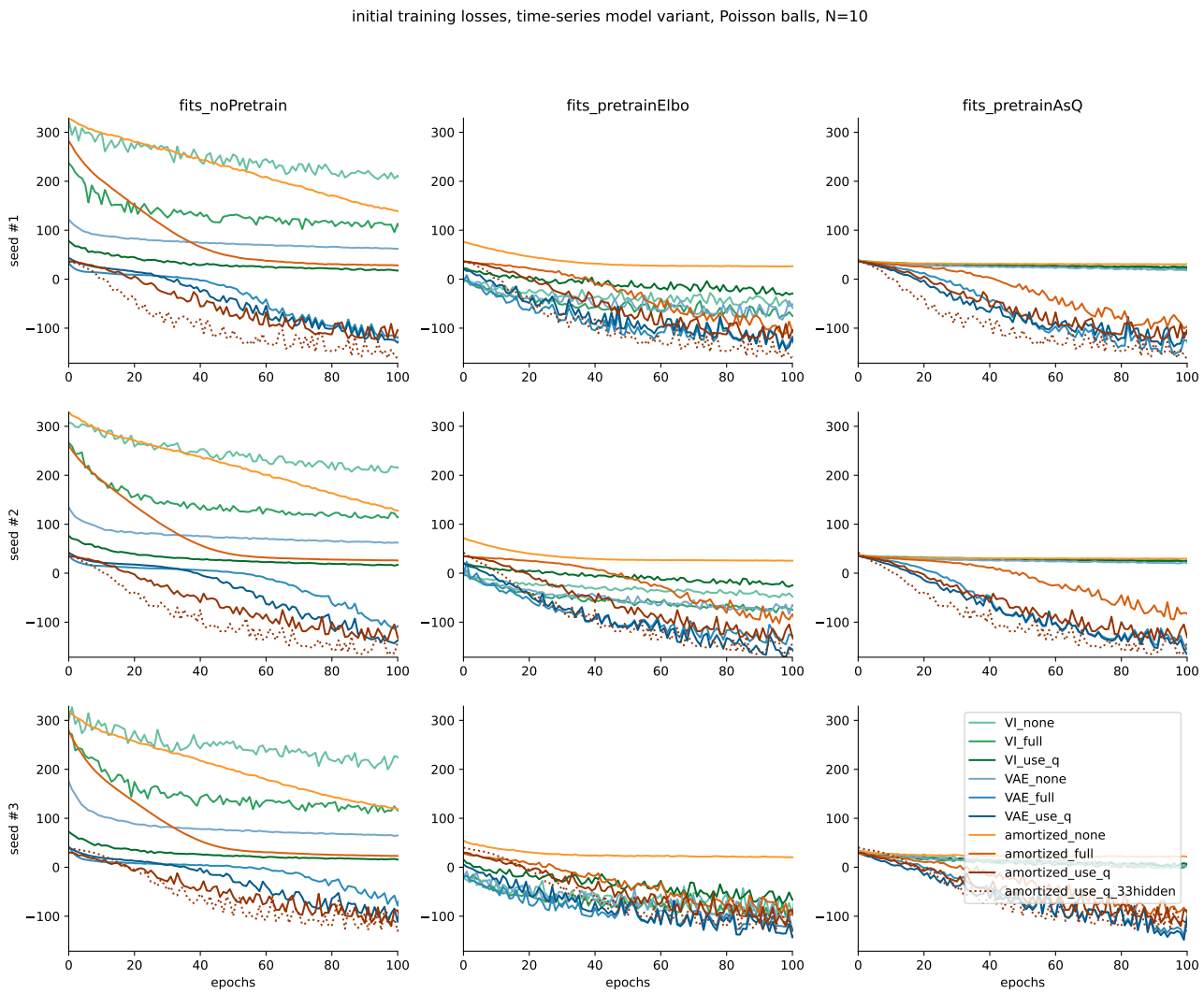

Figure 3: Zoom in on initial training losses (negative average free energy, up to a constant $T/2\log(2\pi)$) for time-series RPM from different initializations. Data here is $N = 10$ Poisson ball samples, with 3 different random seeds for both model parameters and data shown from top to bottom. Left to right: random initialization for all of $\theta, \psi, \{\psi_j\}$; pre-trained $\psi, \{\psi_j\}$ using ELBO; pre-trained $\psi, \{\psi_j\}$ using $\eta_q = \eta_0 - J\eta_0^{unc} + \sum_j \eta_{\theta_j}^{unc}$ and $\tilde{\eta}_j = \eta_0 - \eta_q$ assumptions. Note how the latter initialization quenches almost all initialization variability except for θ. Also note how the 'minimalist amortised' ('amortized_use_q', brown) learning curves for each seed are identical across initializations. The 'minimalist amortized' variant with 33 hidden units per layer has a different θ compared to all other fits (20 hidden units) and hence behaves differently. Full learning curves in Fig A.3.

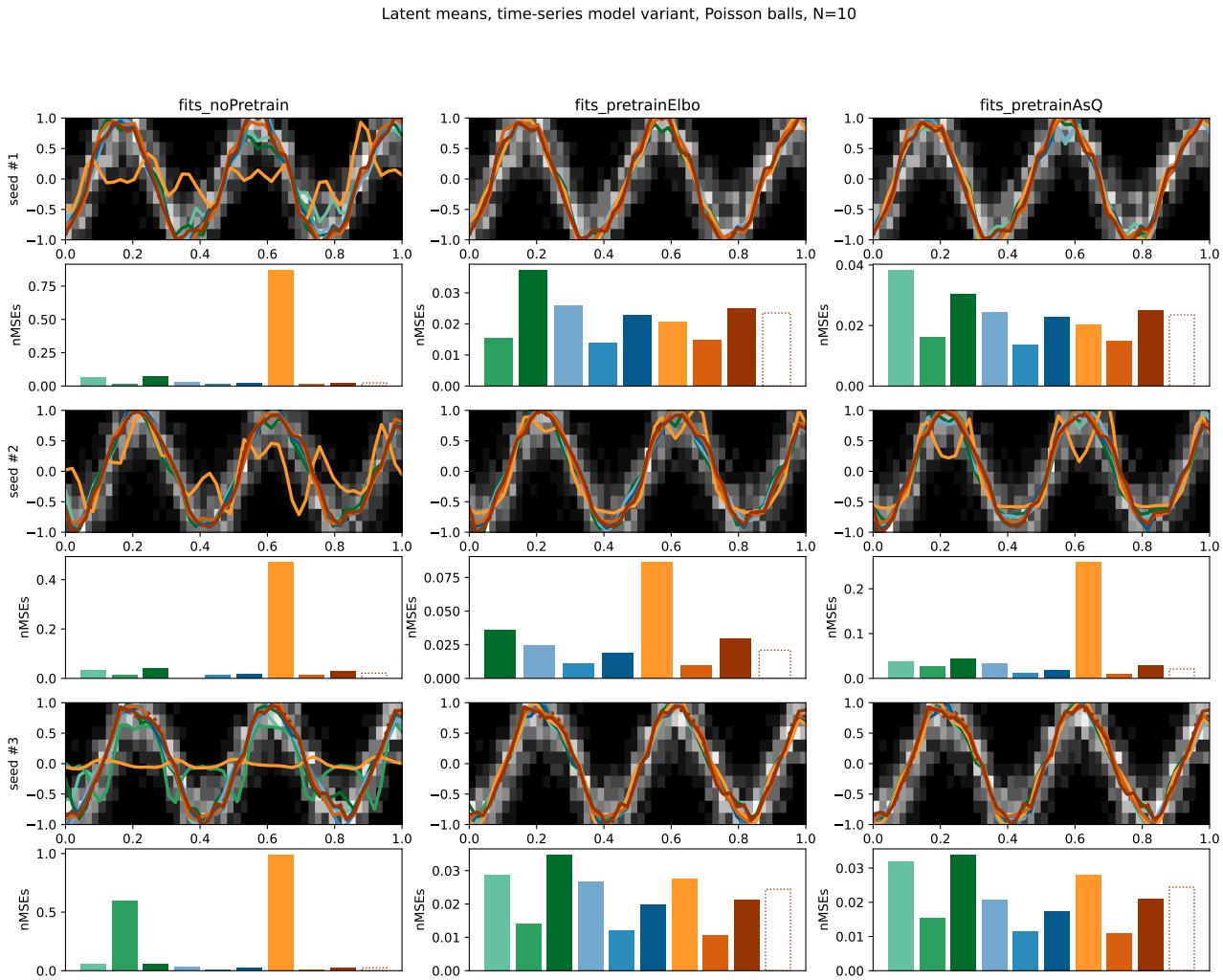

Figure 4: Posterior means $\mathbb{E}_{q(\mathcal{Z}|\mathcal{X})}[\mathcal{Z}] = \nabla\Phi(\eta_q(\mathcal{X}))$ obtained from RPMs after training with different RPM training frameworks for $N = 10$ Poisson ball datapoints. Colors as in Fig 2, with dashed line marking the 'minimalist amortized' variant with 33 hidden units. Left to right: different initialisations (same as Fig 3), top to bottom: different seeds. Top rows: single training data example point overlayed with posterior means, bottom rows: normalized MSE between posterior means and true latents averaged across all $N = 10$ training data points. Missing posterior means/ bars (such as in the central panel) indicate a broken model fit due to loss divergence/NaN during fit, which throughout all experiments only happened for RPM training frameworks that include non-parametric treatment of either η_q and/or $\tilde{\eta}_j$. Overall, 'full VAE' (medium blue) does best, but 'minimalist amortized' (brown) – with the most model restrictions on η_q and $\tilde{\eta}_j$ – does okay, as well.

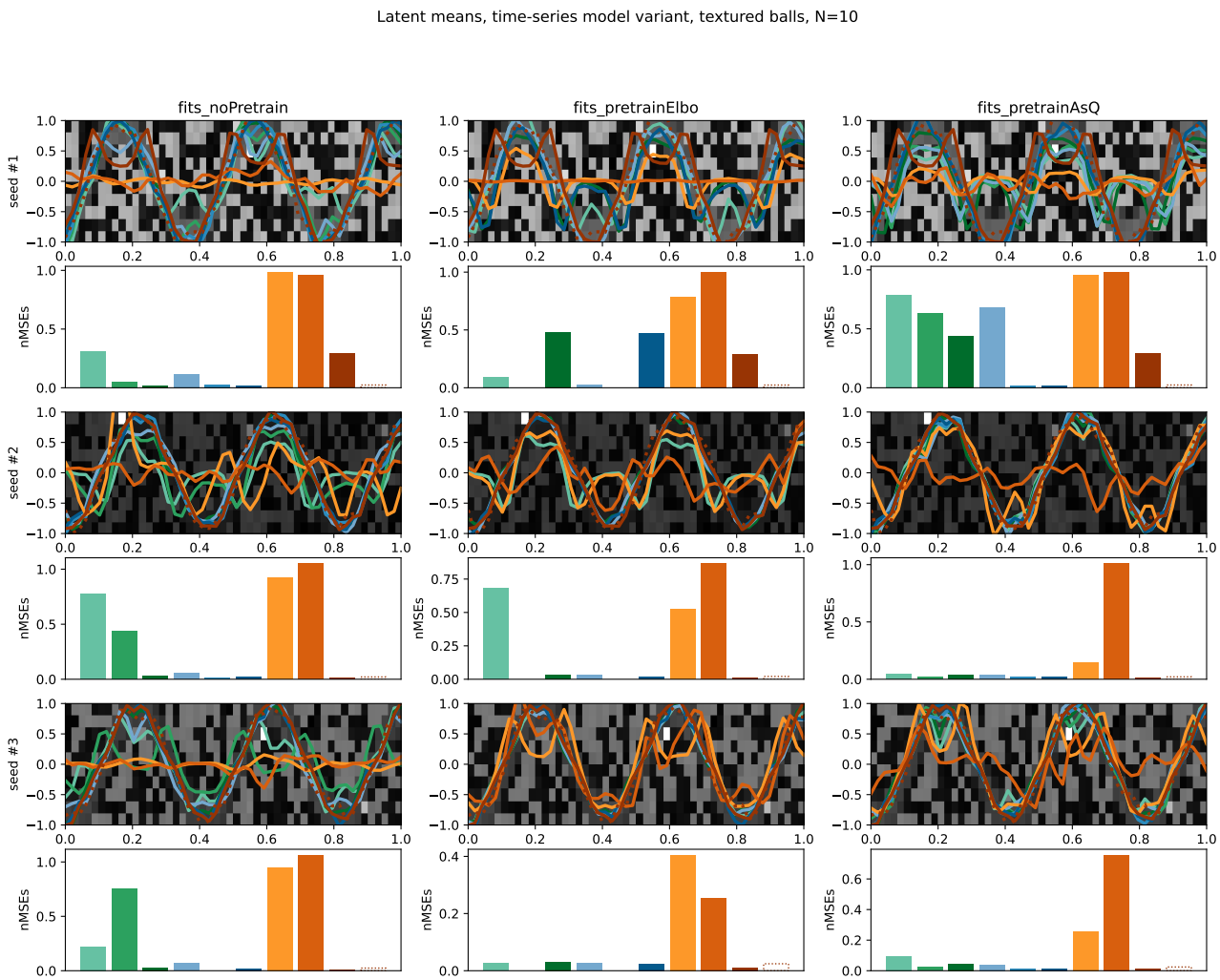

Figure 5: Posterior means $\mathbb{E}_{q(\mathcal{Z}|\mathcal{X})}[\mathcal{Z}] = \nabla\Phi(\eta_q(\mathcal{X}))$ obtained from final fits of different RPM training frameworks for $N = 10$ **textured** ball datapoints. Same as Fig 4, again with brown dashed line marking the 'minimalist amortized' variant with 33 hidden units.

3.2 Larger datasets ($N = 100$) for the 'minimalist amortized' initialization.

We compare RPM training frameworks for the Poisson and textured bouncing ball experiments with $N = 100$. This time, all fits are with pre-trained (inner) recognition models to match the 'minimalist amortized' framework with $\eta_q = \eta_0 - J\eta_0^{unc} + \sum_j \eta_j^{unc}$ and $\tilde{\eta}_j = \eta_0 - \eta_q$, for approximately identical initial conditions across all RPM training frameworks (right-most column in Figures 3-5). We use 2.000 training epochs at batch-size 16.

We again focus on the time-series RPM formulation – the standard RPM formulation (ignoring conditional independence across time) overall works much worse, see appendix A.7 for a comparison of different RPM training frameworks also on the non-time-series RPM formulation with $N = 10$ and $N = 100$.

In summary, on Poisson balls we again find 'full VAE' to work best, but the 'minimalist amortized' works okay. The clear advantage of the 'minimalist amortized' variant that we saw on $N = 10$ textured ball samples earlier – where it was the only competitive framework that didn't overfit on the small dataset size – is gone for $N = 100$, but it still works best across all seeds on the textured balls.

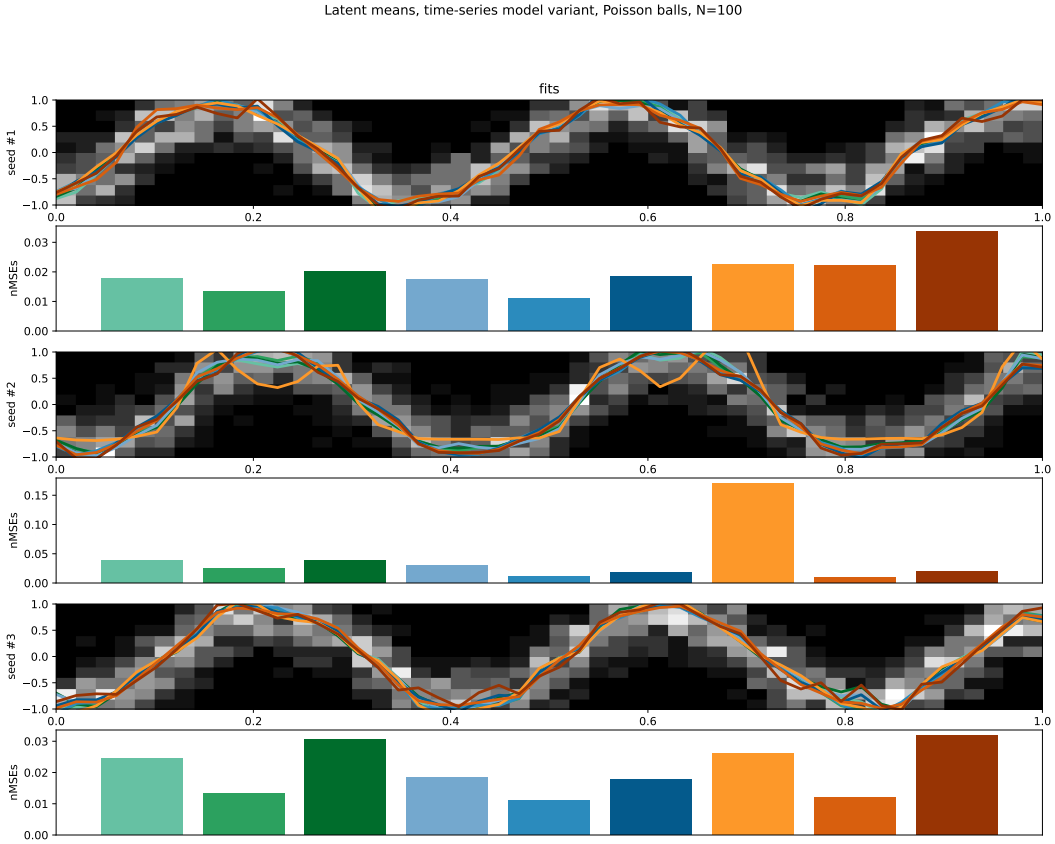

Figure 6: Posterior means $\mathbb{E}_{q(\mathcal{Z}|\mathcal{X})}[\mathcal{Z}] = \nabla\Phi(\eta_q(\mathcal{X}))$ obtained from final fits of different RPM training frameworks for $N = 100$ Poisson ball datapoints. Colors as in Fig 2. Top rows: single training data example point overlay with posterior means, bottom rows: normalized MSE between posterior means and true latents averaged across all $N = 100$ training data points. All results are okay. Surpringly, results for RPM training frameworks with non-parametric natural parameters (all green, as well bright blue/bright orange) are also good despite the training losses not seemingly converge yet (cf. Fig A.14). Overall best is 'full VAE'.

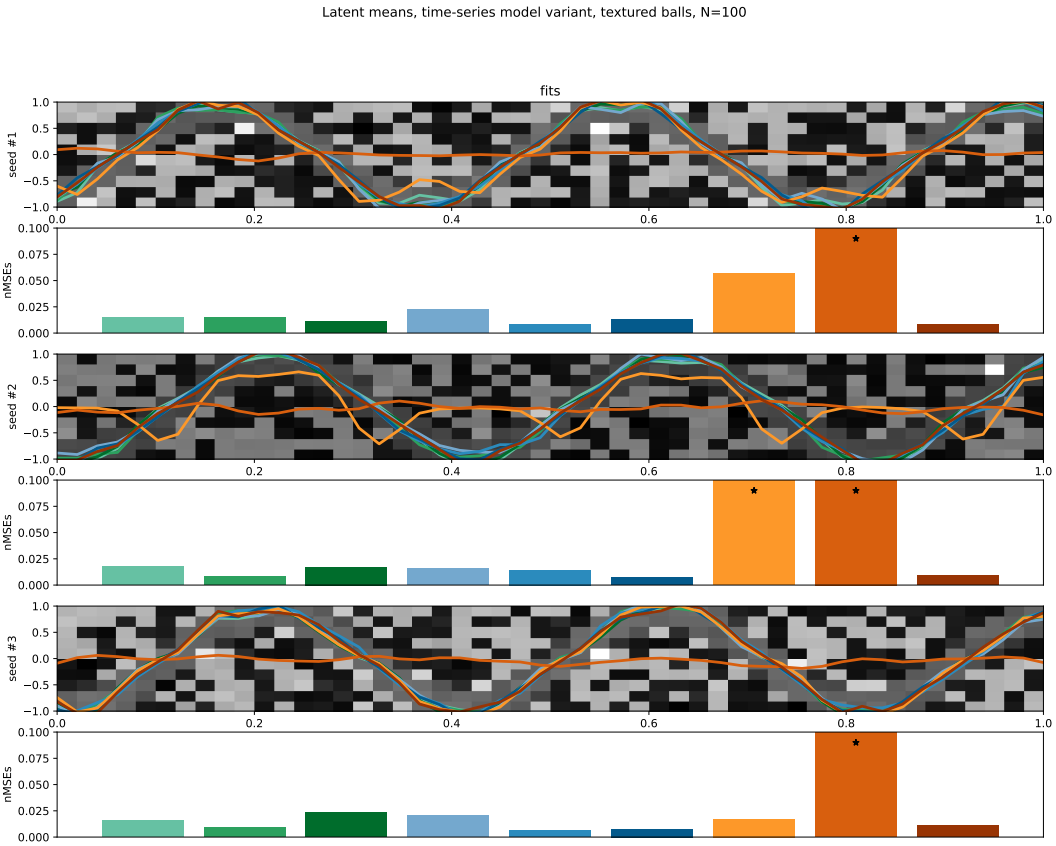

Figure 7: Posterior means $\mathbb{E}_{q(\mathcal{Z}|\mathcal{X})}[\mathcal{Z}] = \nabla\Phi(\eta_q(\mathcal{X}))$ obtained from final fits of different RPM training frameworks for $N = 100$ textured ball datapoints. Colors as in Fig 2. Top rows: single training data example point overlayed with posterior means, bottom rows: normalized MSE between posterior means and true latents averaged across all $N = 100$ training data points. Note how the bars marked with asterisks are truncated at 0.1 nMSE to showcase differences of the better-performing training frameworks. 'minimalist amortized', 'cheap VAE' and 'full VAE' frameworks work best.

A Appendix

A.1 Experimental setup

In both experiments, $\mathcal{X} \in \mathbb{R}^{10\times 50}$ with $\mathcal{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_J], J = 10$ and $\mathbf{x}_j \in \mathbb{R}^T, T = 50$. We try dataset sizes $N = 10$ and $N = 100$ – the original plan was to also test $N = 1000$, but in the current implementation, a tensor of size $N \times$ batch-size $\times\, J \times T \times 2$ is created to compute all the normalizers $F_{\theta_j}(\mathcal{Z}_t)$ in one go in parallel - for $N = 1000$, this implementation rips my workstation's CPU (!) memory. The alternative of serializing that tensor does not sound like fun compute times... better to first look again into subsampling N for the computation of $F_{\theta_j}(\mathcal{Z})$. Note that Walker et al. (2023) used $N = 50$.

A.2 RPM setup

All RPMs are with T-dimensional Gaussian recognition factors and priors, i.e. $p_\theta(\mathcal{Z}) = \mathcal{N}(\mu_0, \Sigma_0)$ with $\mu_0 = 0$, $\Sigma_0^{kl} = \exp(\gamma|t_k - t_l|^2)$ given by RBF kernel with bandwitdh γ and temporal locations $t_k, t_l \in [0, 1]$. Factors $f_{\theta_j}(\mathcal{Z}|\mathbf{x}_j) = \mathcal{N}(\mu_j(\mathbf{x}_j), \Sigma_j(\mathbf{x}_j))$ with $\mu_j(\mathbf{x}_j) = [\mu_j(\mathbf{x}_{j1}), \ldots, \mu_j(\mathbf{x}_{jT})]$ and diagonal $\Sigma_j(\mathbf{x}_j)$, $[\Sigma_j(\mathbf{x}_j)]_{tt} = \sigma_j^2(\mathbf{x}_{jt})$.

For recognition factors and (inner) recognition models, we use 3-layer ReLU conv-Nets with 20 hidden units.

Note that Walker et al. (2023) according to personal conversation treated the data as $J = 1$ (i.e. $T = 50$ independent observations $\mathbf{x}_t \in \mathbb{R}^J$, ignoring the conditional independence across the 10 rows of $\mathcal{X}$).

[I should say there currently is an issue with learning the prior parameter γ, which does obtain a valid gradient (via auto-differentiation), but doesn't seem to budge much from initialization (initalized at $\gamma = 1000$, changes by some 5% over 10k gradient steps, explaining in some parts the non-smooth posterior means I get – the other explanation being that I use $J = 10$ instead of Hugo's $J = 1$, meaning that his recognition factors $f_\theta(\mathcal{Z}_t|\mathbf{x}_{jt})$ actually saw a full column of $\mathcal{X} \in \mathbb{R}^{10\times 50}$ and hence get to know where exactly the latent is in that column, but mine never do).]

A.3 Parameter counts across RPM learning frameworks

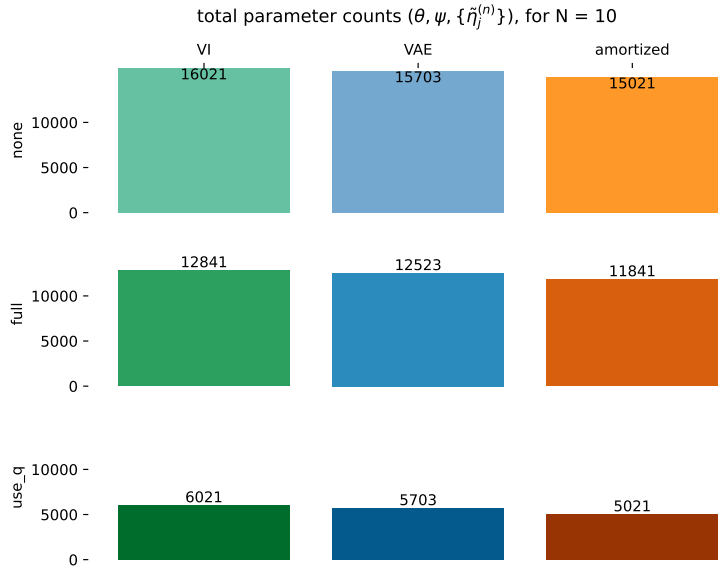

Figure A.1: RPM framework total parameter counts (across all of (θ, ψ, ψ_j)) for both bouncing ball experiments with $N = 10$. Labels are internal names in the code for the same model variants as in Fig 2 (same colors). Note that the costs in parameter counts for the parametric methods are so comparatively low here because the models are fully convolutional along time ($\eta_{\theta_j}(\mathbf{x}_{jt})$ shares parameters over all $t = 1, \ldots, T$).

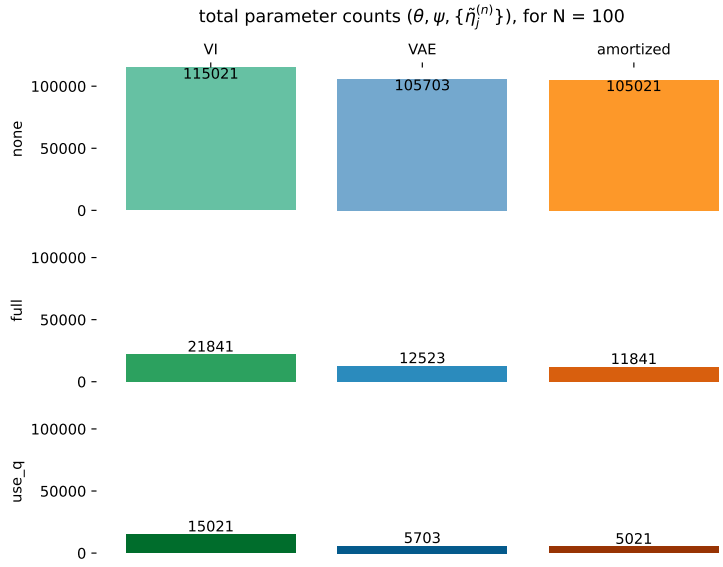

Figure A.2: RPM framework total parameter counts for both bouncing ball experiments with $N = 100$. The bottom-and right-most four frameworks do not change their parameter count since they are fully amortized.

A.4 Additional results for $N = 10$ with various initialiations (but few epochs)

A.4.1 Poisson bouncing balls

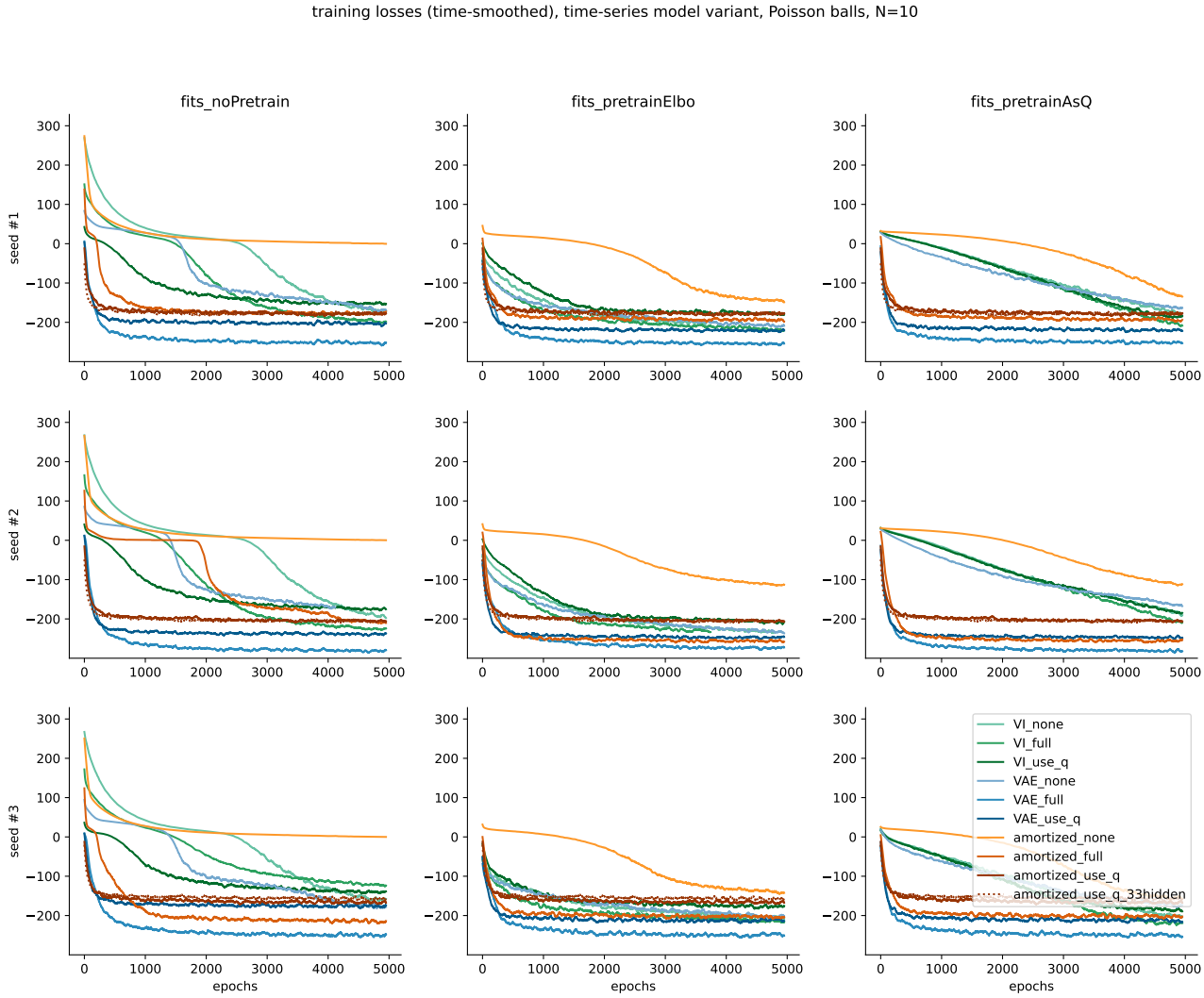

Figure A.3: Same as in Fig 3, but full training curves across 5000 gradient steps (=epochs with batch-size 8). Mini-batch errors are smoothed with a running average over 50 gradient steps for visual clarity. Note how in each case, four RPM training frameworks converge within ≈ 1000 steps: VAE and 'amortized', with parametric inner variational parameters ('use_q' or 'full'). All other RPM frameworks – including non-parametric η_q and/or non-parametric $\tilde{\eta}_j$ – require much longer, in line with their training signals being much more sparse.

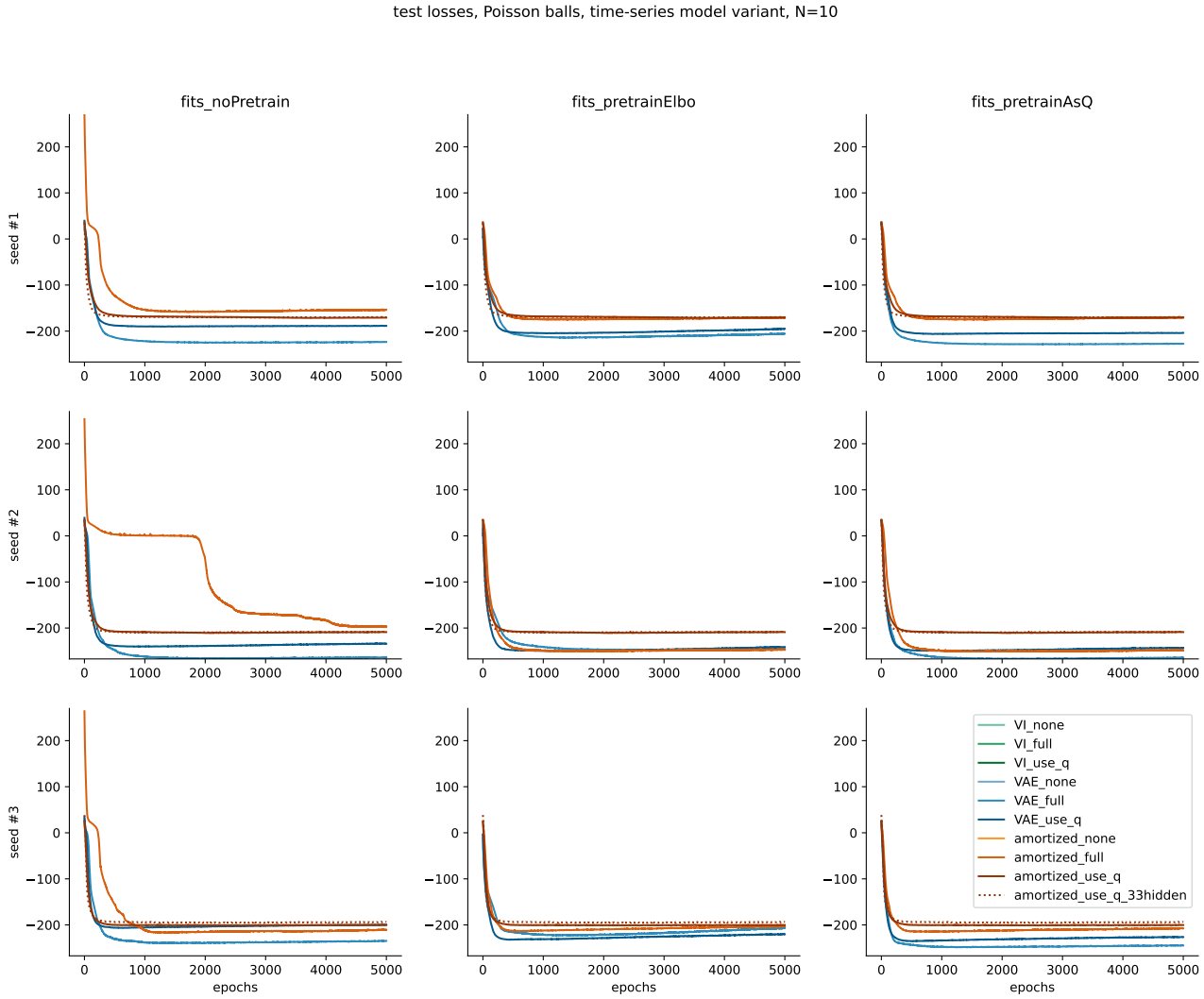

Figure A.4: Same as in Fig 3, but full test loss over a separate test set with $N_{\text{test}} = N = 10$ across 5000 gradient steps (=epochs with batch-size 8), for the four fully amortized RPM training frameworks (i.e. no learning of extra parameters needed to evaluate free energy on test data points). For the Poisson ball experiment, only mild overfitting occurs only for the 'full VAE' variant, and the ordering of final test losses across RPM training frameworks follows the ordering of training losses in Fig A.3. The ordering roughly follows the number of free parameters, with more parameters yielding better results.

A.4.2 Textured bouncing balls

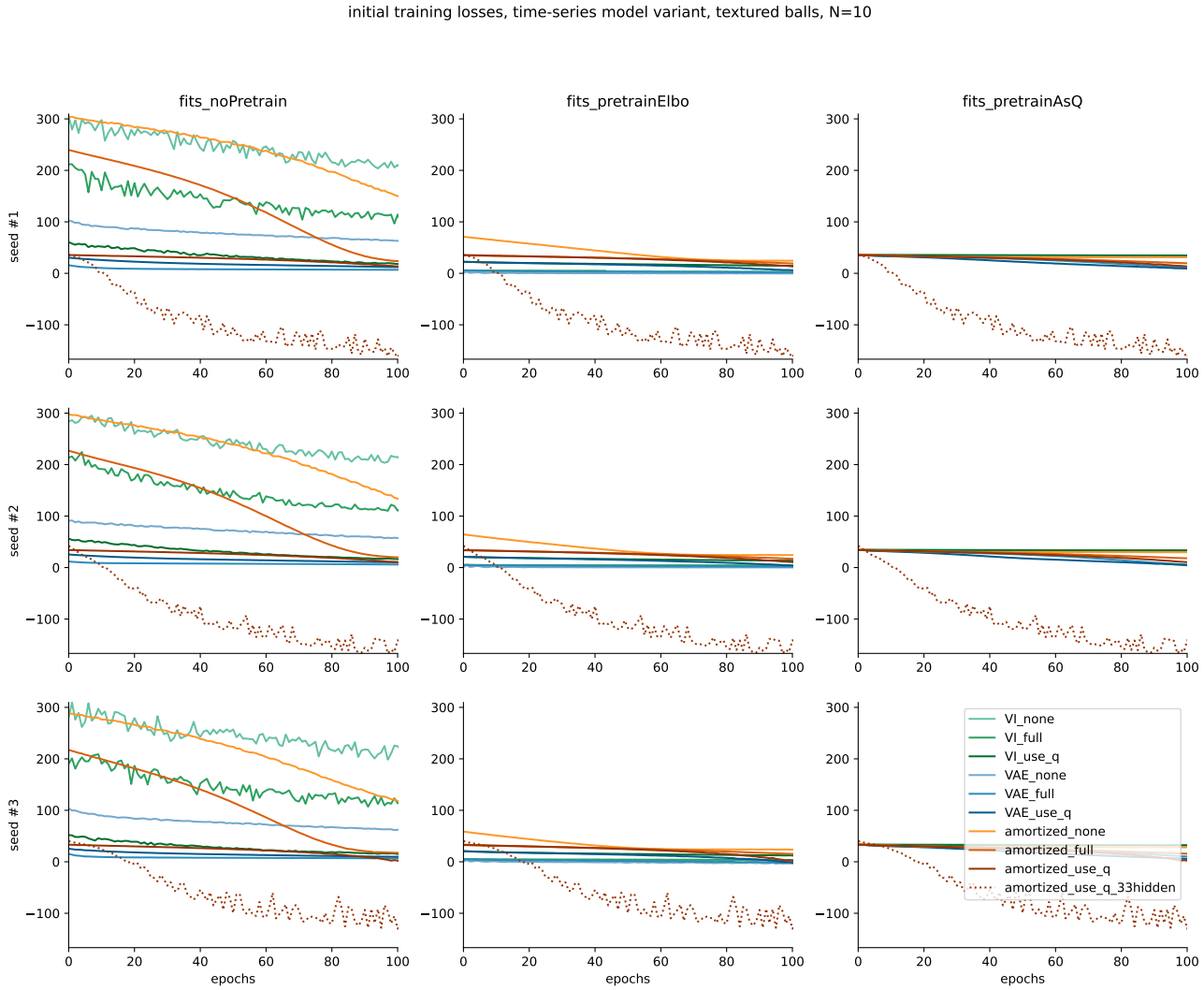

Figure A.5: Same as Fig 3, but for **textured** bouncing balls. Zoom in on initial training losses for time-series RPM from different initializations. Data here is $N = 10$ textured ball samples, for 3 different random seeds.

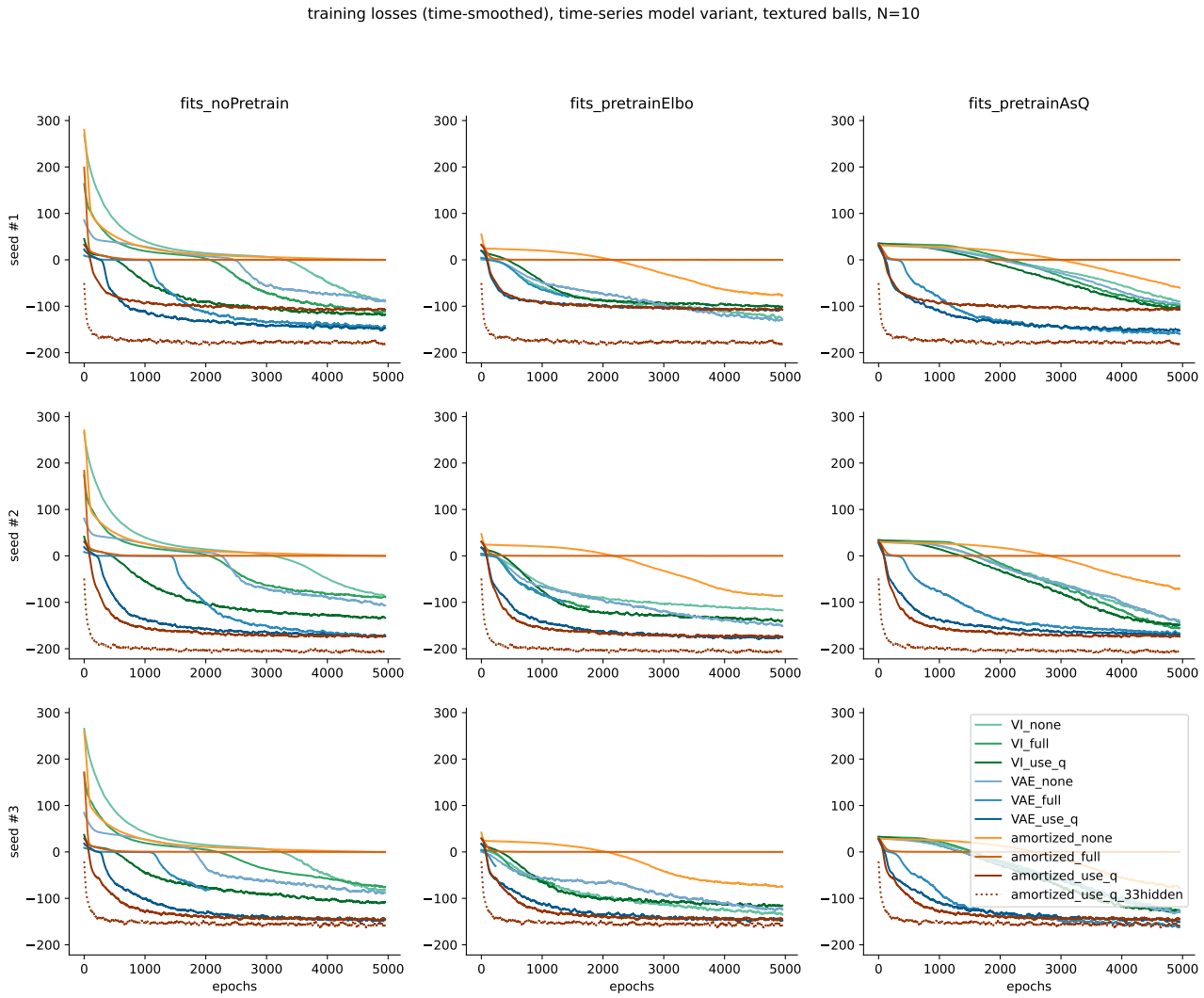

Figure A.6: Full training curves, same as in Fig 3, but **textured** balls, and with smoothed training losses (running average over 50 gradient steps). Note how the 'minimalist amortized' ('amortized_use_q') training framework gives best training errors for this data, with the variant with 33 hiddens working even better. Also note how some fits (green curves, i.e. with non-parametric natural parameters) terminate early due to sudden loss divergence.

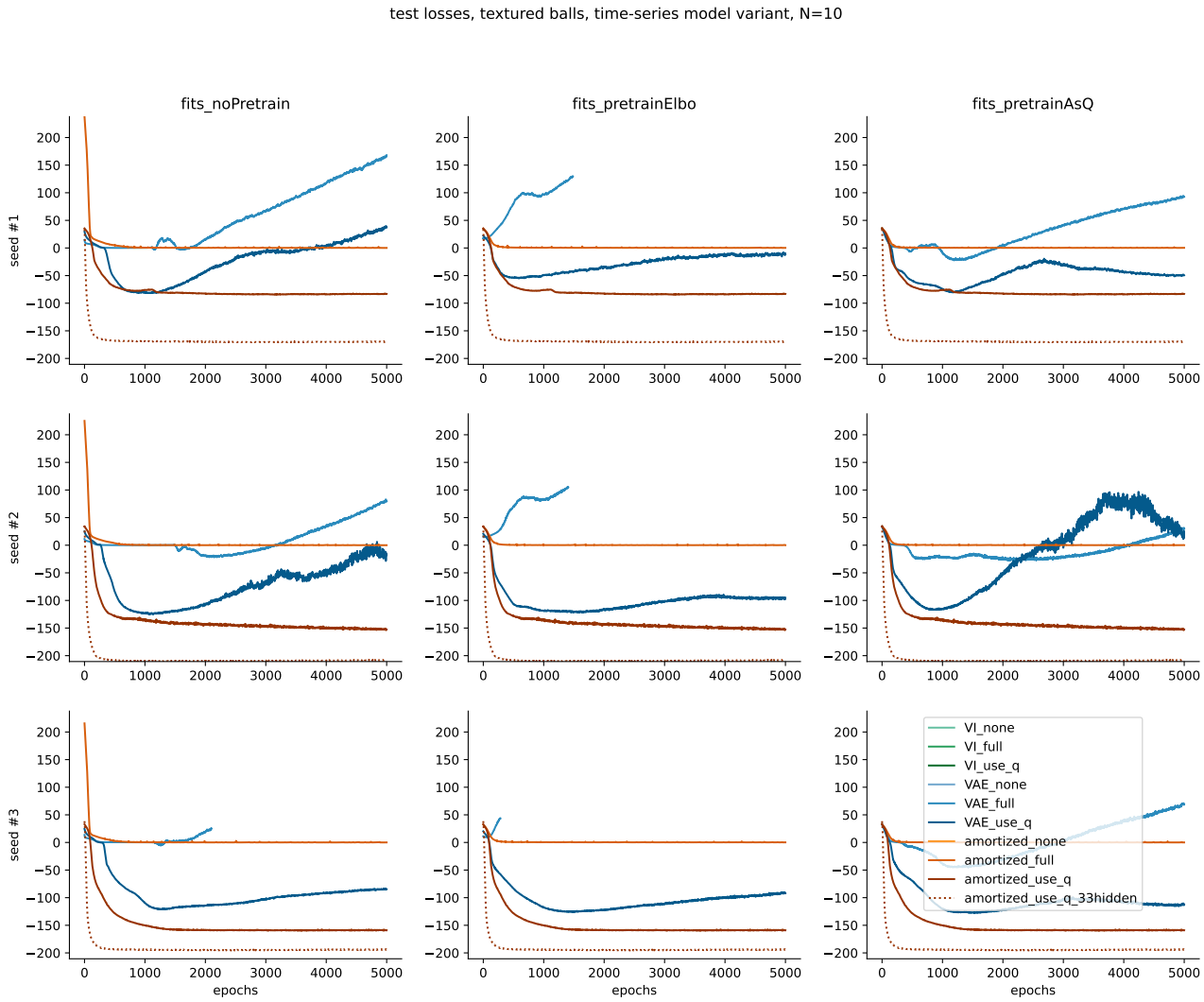

Figure A.7: Test error curves for fully amortized RPM frameworks on $N = 10$. Same as in Fig A.4, but for **textured** ball data. Note how 'full VAE' and 'cheap VAE' ('VAE_use_q') strongly overfit and even become unstable (invalid Gaussian precisions) over the test data.

A.5 Additional results for time-series RPM on $N = 10$ with 'minimalist amortized' initialization and more training epochs

Results for $N = 10$ on longer training fits with 10.000 epochs (cf. the 5.000 epochs from the first batch of comparison with various intializations) at batch-size 8 for the pre-trained recognition models $\eta_q \approx (1 - J)\eta_0 + \sum_j \eta_j$ and $\tilde{\eta}_j \approx \eta_0 - \eta_q$. In summary, the larger number of epochs helps in particular the models with non-parametric natural parameters $\eta_q^{(n)}$ resp. $\tilde{\eta}_j$, as long as the losses don't diverge early in training.

A.5.1 Poisson bouncing balls

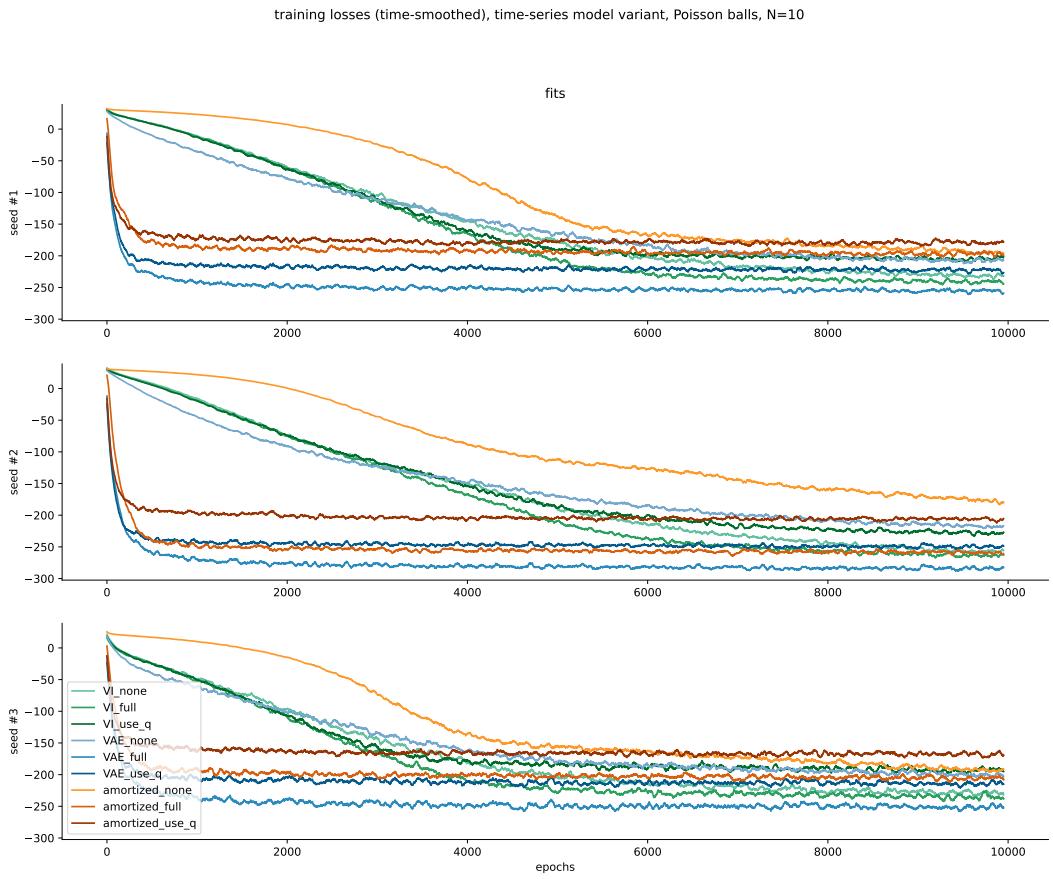

Figure A.8: smoothed training losses on $N = 10$ for Poisson bouncing balls, using a time-series RPM model. Same as in previous section, but longer fits (to give frameworks with non-parametric $\eta_q/\tilde{\eta}_j$ a chance to converge). Number of epochs equals number of gradient steps (batch-size 8).

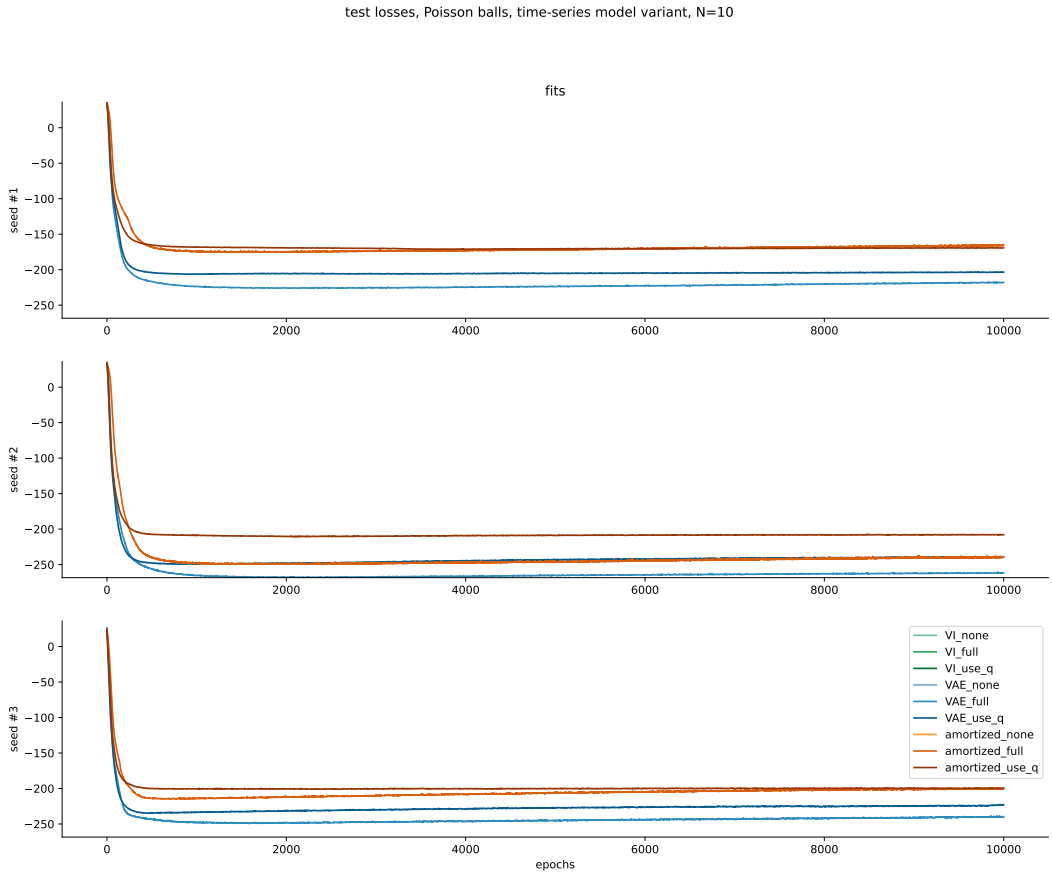

Figure A.9: test losses on $N = 10$ for Poisson bouncing balls for the four fully amortized RPM training frameworks, using a time-series RPM model. Same as in previous section, but longer fits (to give frameworks with non-parametric $\eta_q/\tilde{\eta}_j$ a chance to converge). Number of epochs equals number of gradient steps (batch-size 8).

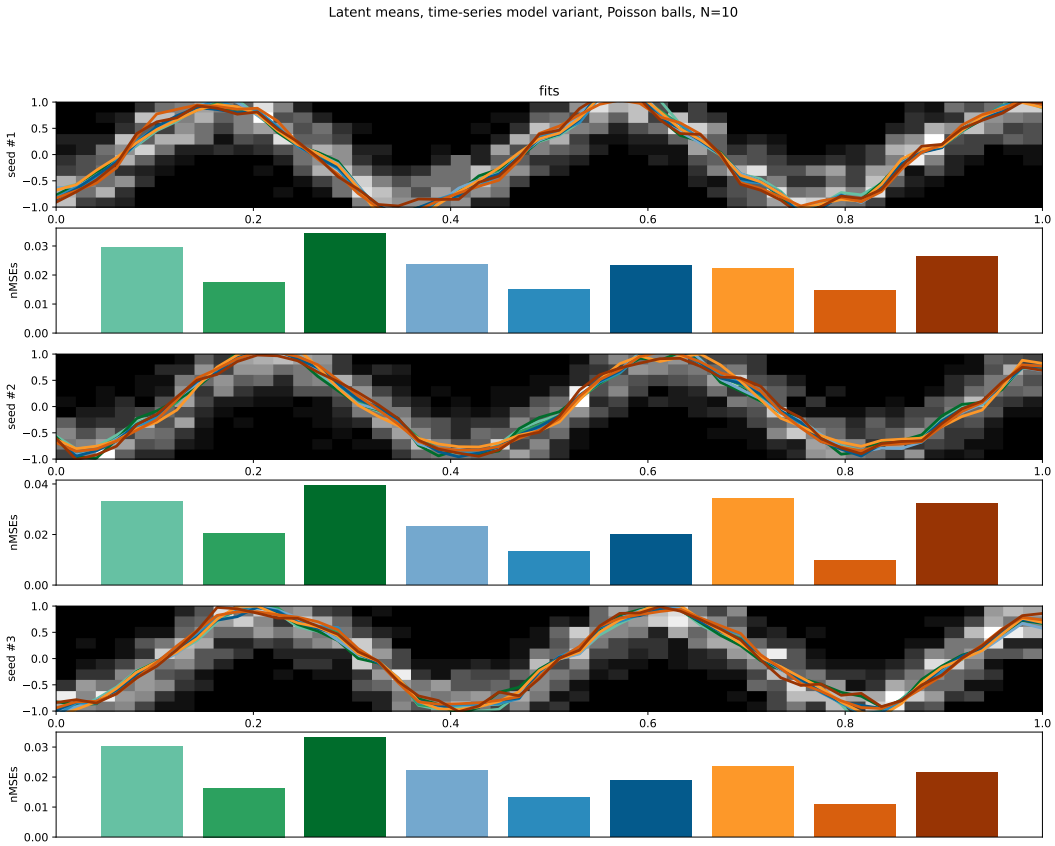

Figure A.10: Posterior means $\mathbb{E}_{q(\mathcal{Z}|\mathcal{X})}[\mathcal{Z}] = \nabla\Phi(\eta_q(\mathcal{X}))$ obtained from final fits of different RPM training frameworks for $N = 10$ Poisson ball datapoints. Colors as in Fig 2. Top rows: single training data example point overlay with posterior means, bottom rows: normalized MSE between posterior means and true latents averaged across all $N = 10$ training data points. The main difference to the shorter runs in the previous section (Fig 4) is the full convergence of the RPM frameworks with non-parametric $\tilde{\eta}_j$, in particular the one with analytic η_q (bright orange, third from right).

A.5.2 textured bouncing balls

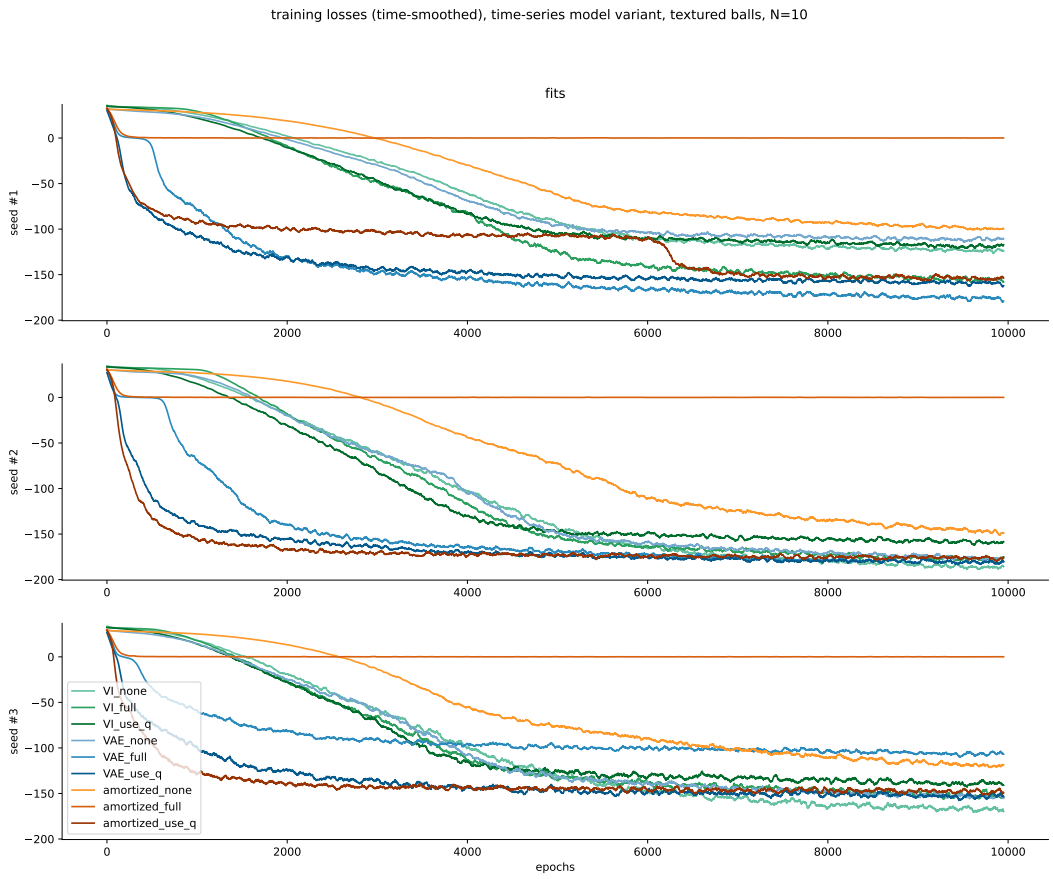

Figure A.11: training losses on $N = 10$ for **textured** bouncing balls, using a time-series RPM model. Same as in previous section, but longer fits (to give frameworks with non-parametric $\eta_q/\tilde{\eta}_j$ a chance to converge). Number of epochs equals number of gradient steps (batch-size 8).

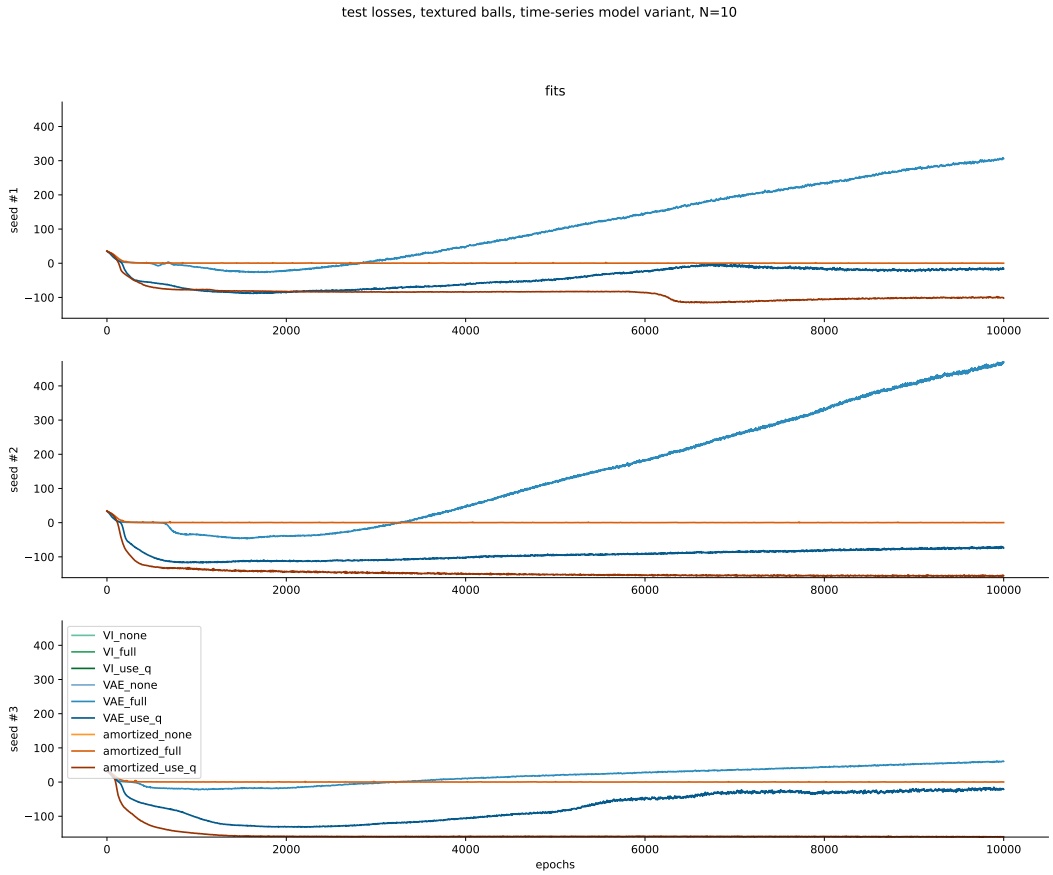

Figure A.12: test losses on $N = 10$ for textured bouncing balls for the four fully amortized RPM training frameworks, using a time-series RPM model. Same as in previous section, but longer fits (to give frameworks with non-parametric $\eta_q/\tilde{\eta}_j$ a chance to converge). Number of epochs equals number of gradient steps (batch-size 8). Note again the clear overfitting of 'VAE'-type frameworks.

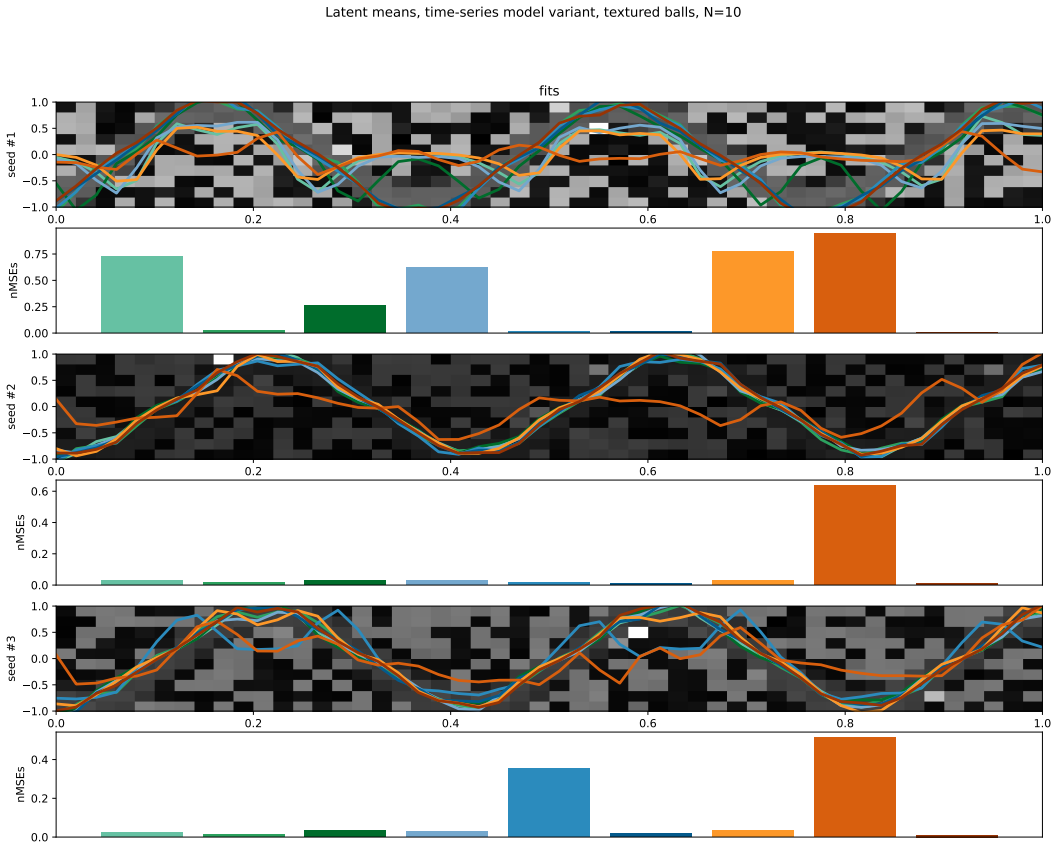

Figure A.13: Posterior means $\mathbb{E}_{q(\mathcal{Z}|\mathcal{X})}[\mathcal{Z}] = \nabla\Phi(\eta_q(\mathcal{X}))$ obtained from final fits of different RPM training frameworks for $N = 10$ textured ball datapoints. Colors as in Fig 2. Top rows: single training data example point overlay with posterior means, bottom rows: normalized MSE between posterior means and true latents averaged across all $N = 10$ training data points. 'minimalist amortized' (brown) overall works best.

A.6 Training and test errors for time-series RPM on $N = 100$ with 'minimalist amortized' initialization

A.6.1 Poisson bouncing balls

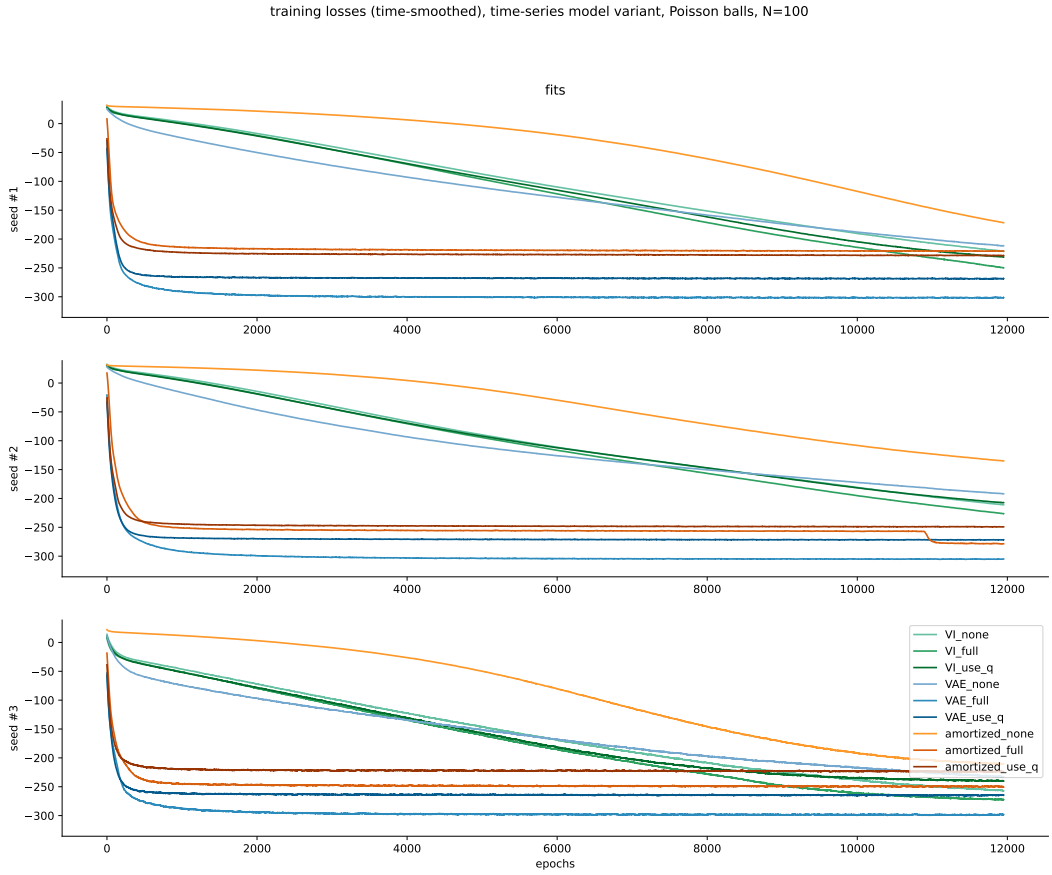

Figure A.14: training losses on $N = 100$ for Poisson bouncing balls, using a time-series RPM model. Training losses are smoothed using a running average over 50 training steps. Note how the training frameworks with non-parametric natural parameters $\eta_q^{(n)}, \tilde{\eta}_j^{(n)}$ fail to converge within the 12000 gradient steps (2000 epochs with batch-size 16), in line with the increased sparsity of the learning signal for non-parametric natural parameters for this larger dataset (cf. Fig A.8).

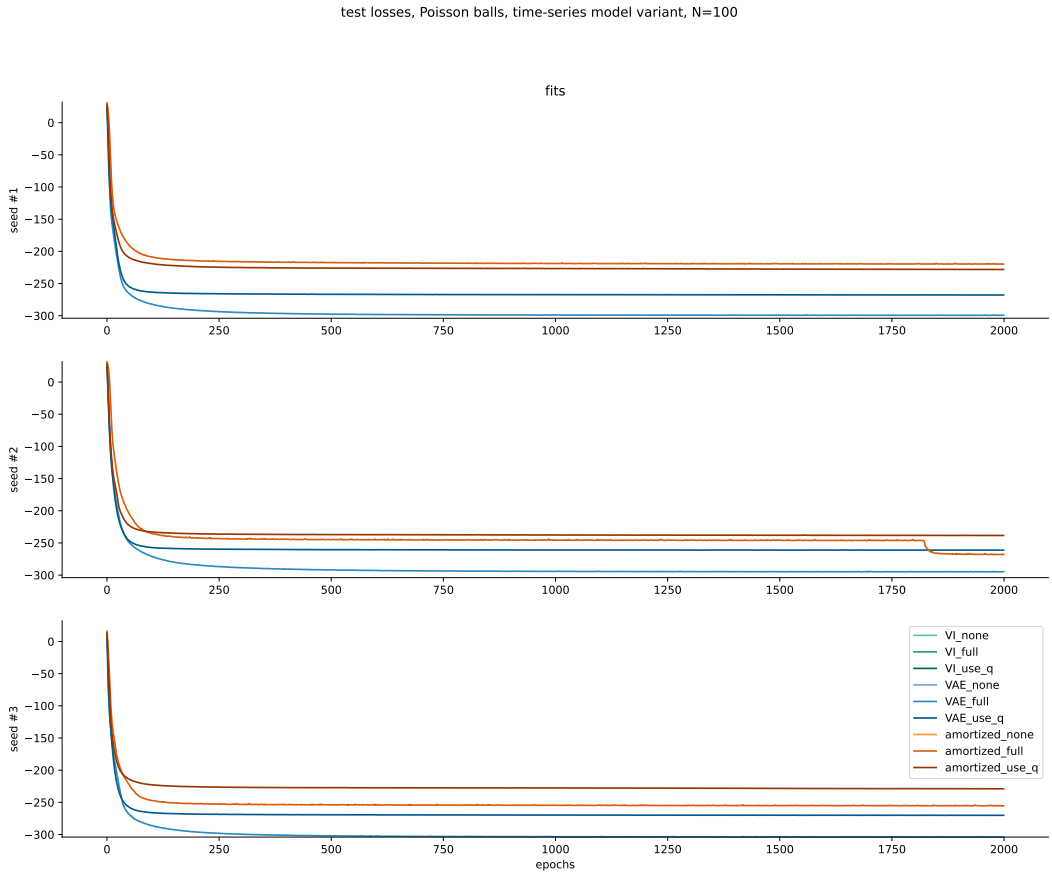

Figure A.15: test losses on a held-out test dataset set with $N_{\text{test}} = N = 100$ for Poisson bouncing balls for the four fully amortized RPM training frameworks, using a time-series RPM model, over 2000 training epochs.

A.6.2 textured bouncing balls

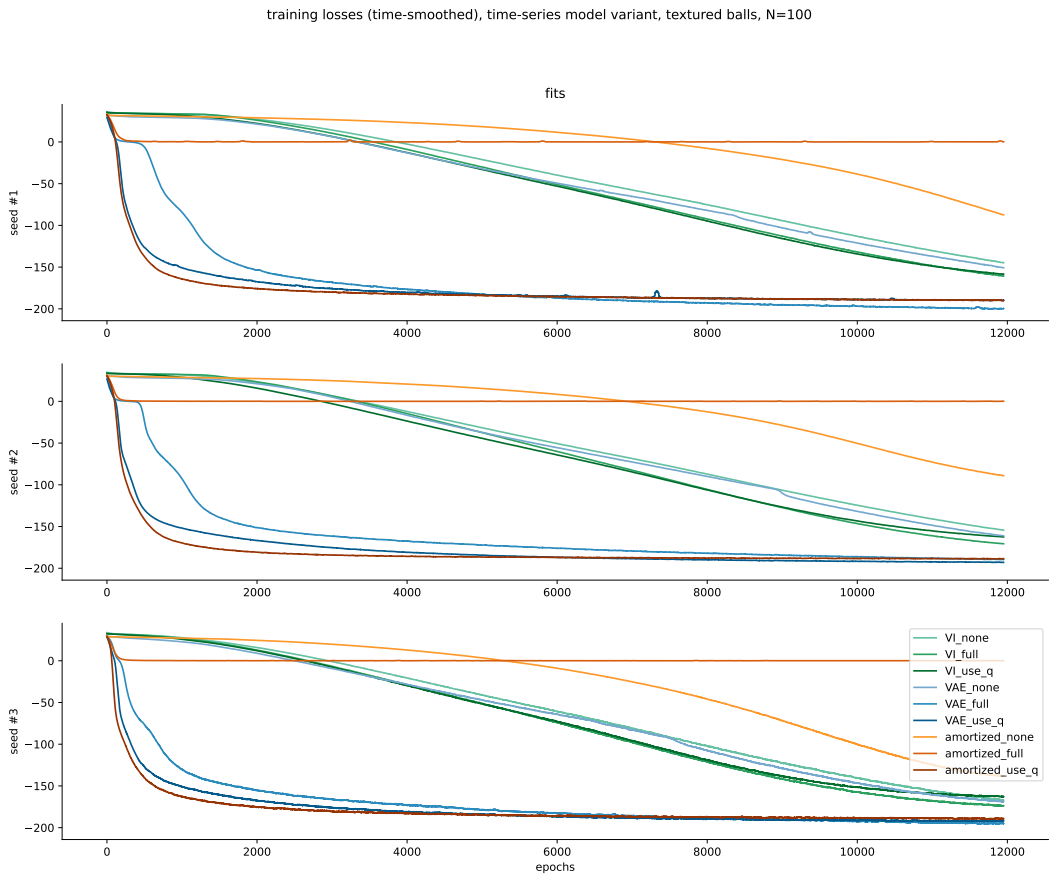

Figure A.16: training losses on $N = 100$ for textured bouncing balls, using a time-series RPM model. Training losses are smoothed using a running average over 50 training steps. Note how the training frameworks with non-parametric natural parameters $\eta_q^{(n)}, \tilde{\eta}_j^{(n)}$ fail to converge within the 12000 gradient steps (2000 epochs with batch-size 16), in line with the increased sparsity of the learning signal for non-parametric natural parameters for this larger dataset (cf. Fig A.11).

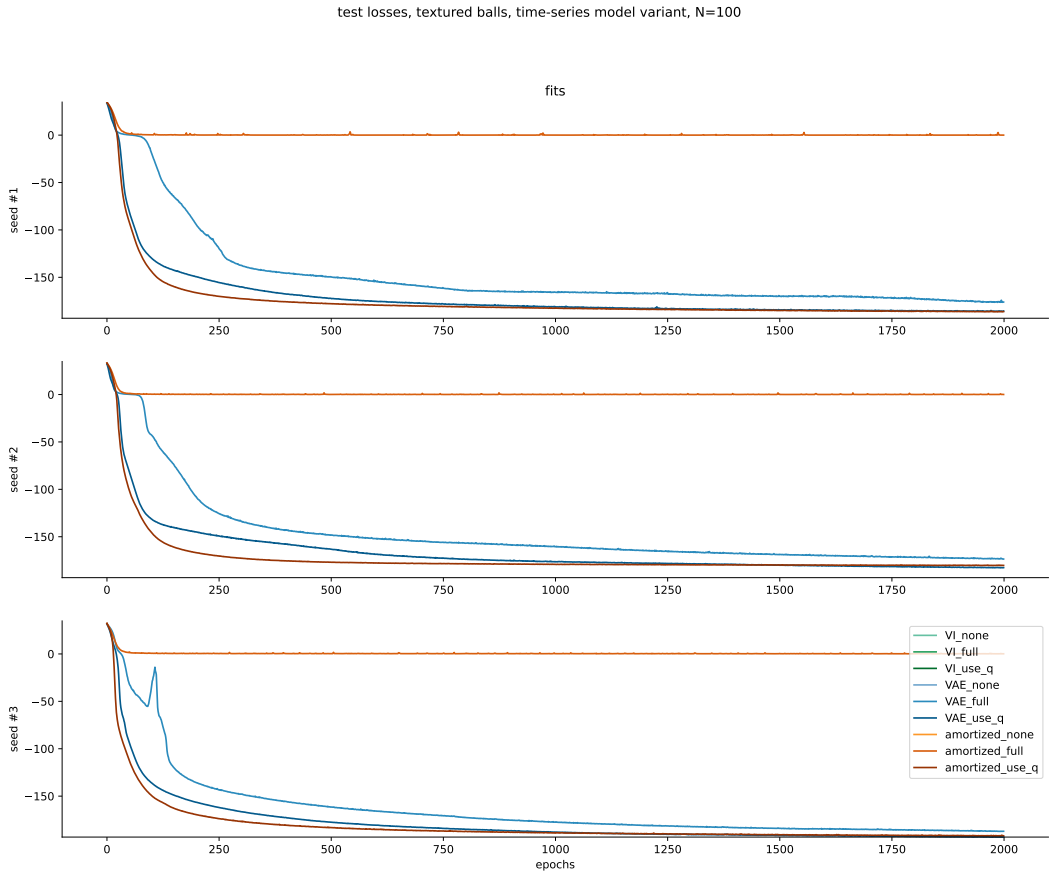

Figure A.17: test losses on a held-out test dataset set with $N_{\text{test}} = N = 100$ for textured bouncing balls for the four fully amortized RPM training frameworks, using a time-series RPM model, over 2000 training epochs. Compared to $N = 10$ (Fig A.12), 'VAE' variants do not visibly overfit and approach the performance of the 'minimalist amortized' variant, which however converges noticeably faster.

A.7 Results for standard (non-time-series) RPM model on $N = 10$ and $N = 100$ for 'minimalist amortized' initialization

Results for the non-time-series RPM (ignoring temporal conditional independences) are considerably worse.

First of, non-parametric $\eta_q^{(n)}$ lead to loss divergence after a few hunded gradient steps – potentially collapse of posterior variances on individual data points (I didn't checkpoint fits, so can't check...).

Fits that did converge show posterior latent means for which it is hard to assert systematic relationships to the true latent at all (should check posterior variances), at least for $N = 10$ and for most fits even at $N = 100$. From visual inspections, the 'minimalist amortized' model seems to still capture best the sinusiodal movement in latent space, though this does not show in normalized MSEs (indeed they're higher than e.g. for 'full VAE').

A.7.1 standard (non-time-series) RPM on Poisson balls

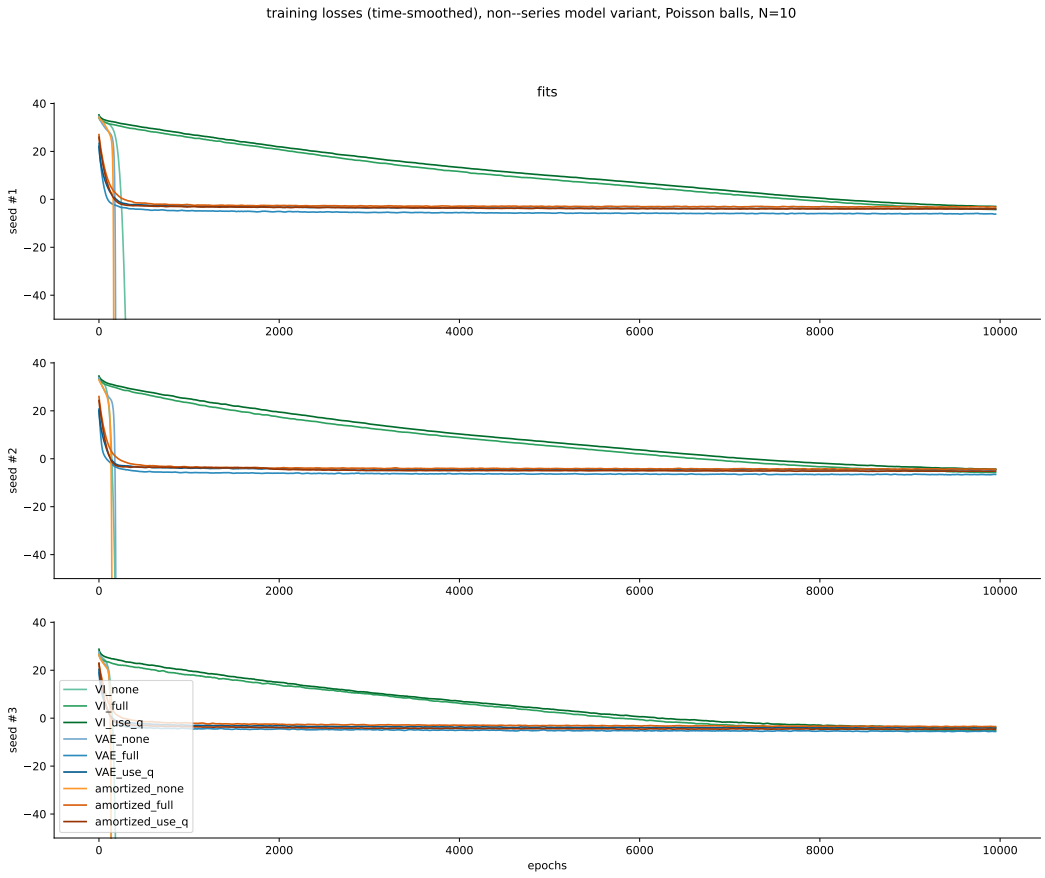

Figure A.18: training losses on $N = 10$ for Poisson bouncing balls, using a standard (non-time-series) RPM model. Number of epochs equals number of gradient steps (batch-size 8). All models with non-parametric natural parameters $\eta_q^{(n)}$ for the recognition model $q(\mathcal{Z}|\mathcal{X}^{(n)})$ collapsed to negative infinite loss within a few hundred gradient steps.

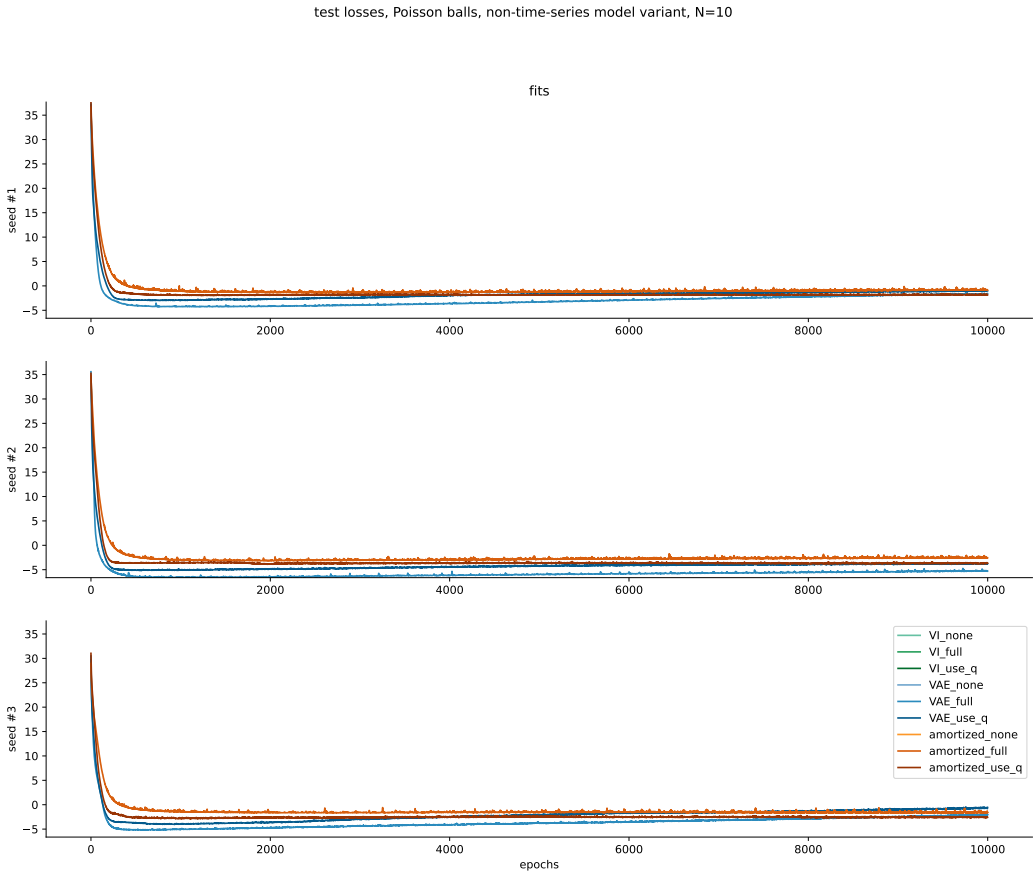

Figure A.19: test losses on $N = 10$ for Poisson bouncing balls for the four fully amortized RPM training frameworks, using a non-time-series RPM model. Number of epochs equals number of gradient steps (batch-size 8). We again see mild overfitting for VAE-type RPM training frameworks, in particular 'full VAE', which overall still gives best test losses.

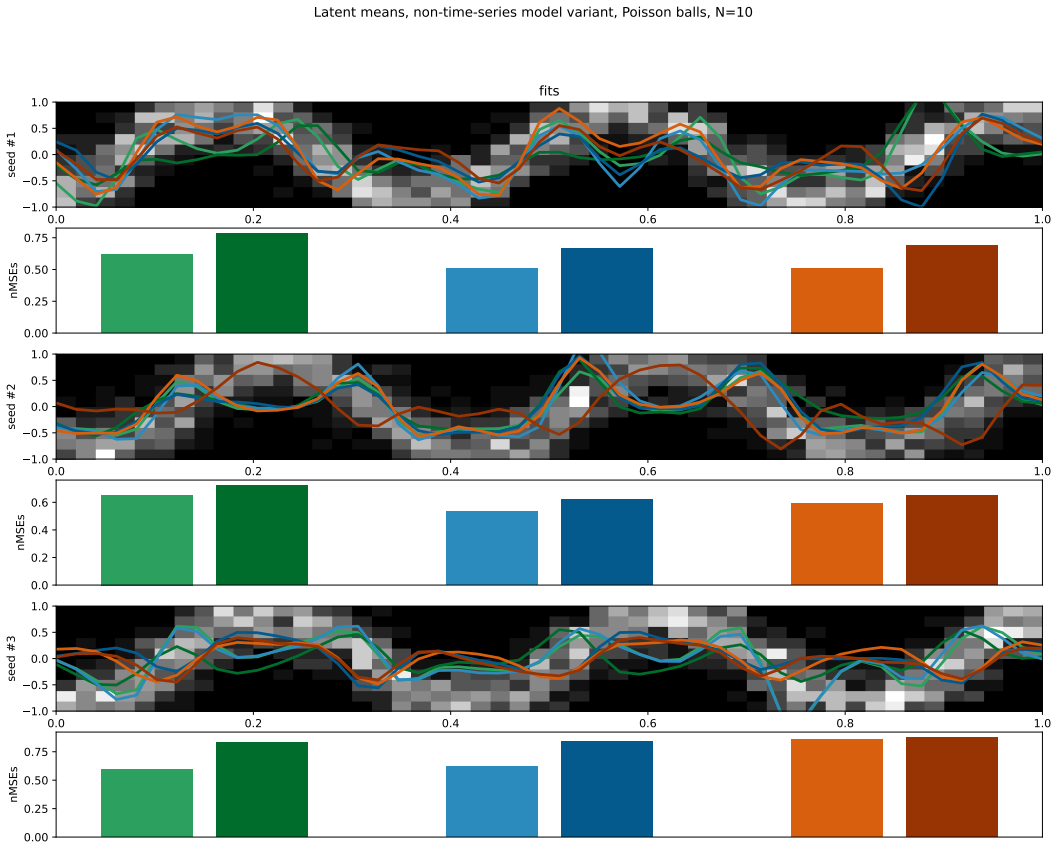

Figure A.20: Posterior means $\mathbb{E}_{q(\mathcal{Z}|\mathcal{X})}[\mathcal{Z}] = \nabla\Phi(\eta_q(\mathcal{X}))$ obtained from final fits of different RPM training frameworks for $N = 10$ Poisson ball datapoints, for non-time-series RPM. Colors as in Fig 2. Top rows: single training data example point overlay with posterior means, bottom rows: normalized MSE between posterior means and true latents averaged across all $N = 10$ training data points. Latents are throughout much worse than for time-series RPM. Fits with non-parametric η_q not included since they all broke (and I neglected to checkpoint the last working update step before the weights turned NaN...).

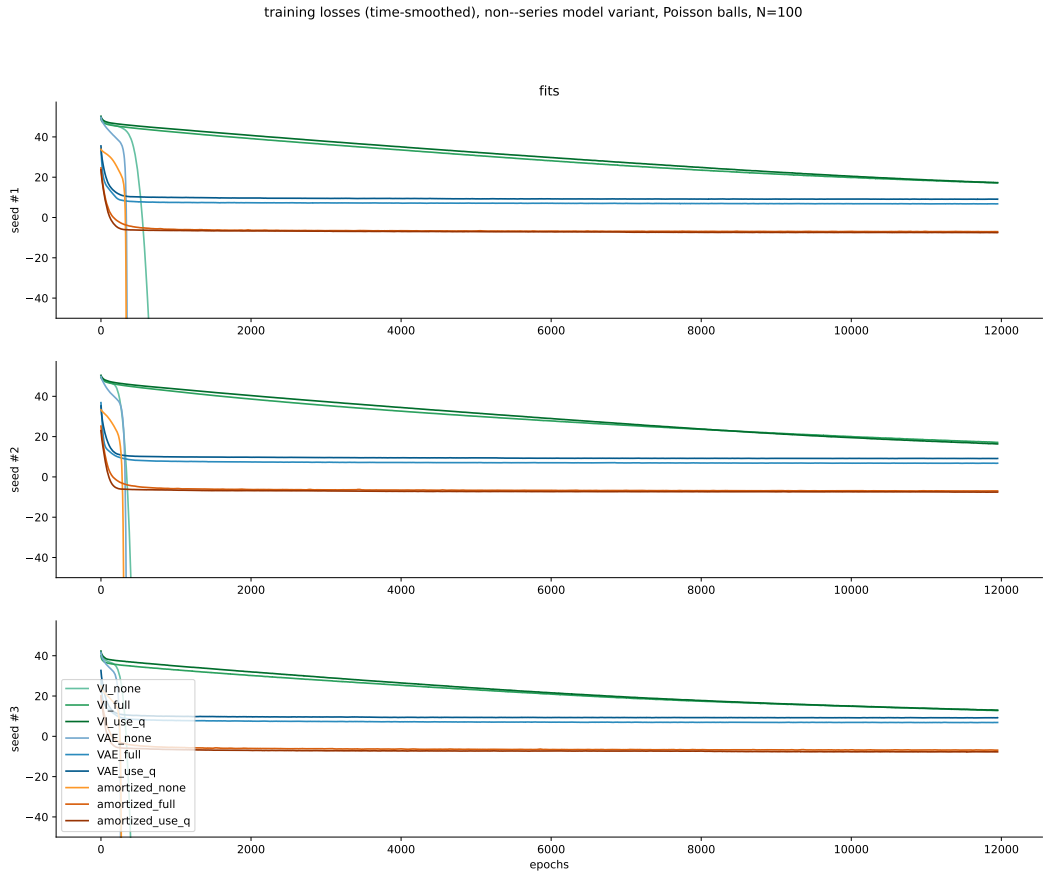

Figure A.21: training losses on $N = 100$ for Poisson bouncing balls, using a non-time-series RPM model. Training losses are smoothed using a running average over 50 training steps. Note how the training frameworks with non-parametric recognition parameter $\eta_q^{(n)}, \tilde{\eta}_j^{(n)}$ again diverge to negative infinite loss. Training frameworks with non-parametric $\tilde{\eta}_j^{(n)}$ but parametric $\eta_q(\mathcal{X})$ stay stable, but fail to converge within 12.000 training steps, again presumably due to the sparser training signal on the $N = 100$ training set (cf. $N = 10$ in Fig A.18).

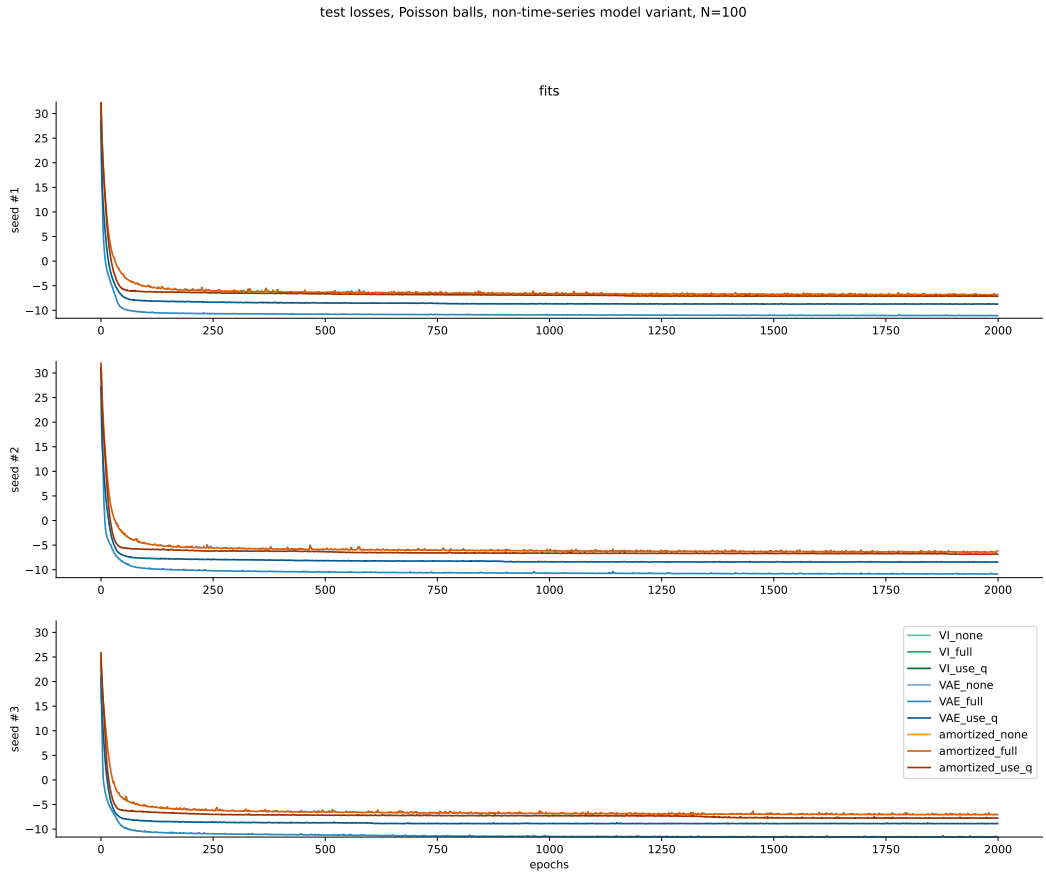

Figure A.22: test losses on a held-out test dataset set with $N_{\text{test}} = N = 100$ for Poisson bouncing balls for the four fully amortized RPM training frameworks, using a non-time-series RPM model, over 2000 training epochs. Note how there is no visible overfitting, but the performance ordering is still flipped ('full VAE' best) compared to the training errors in Fig. A.21 (where 'minimalist amortized' is best).

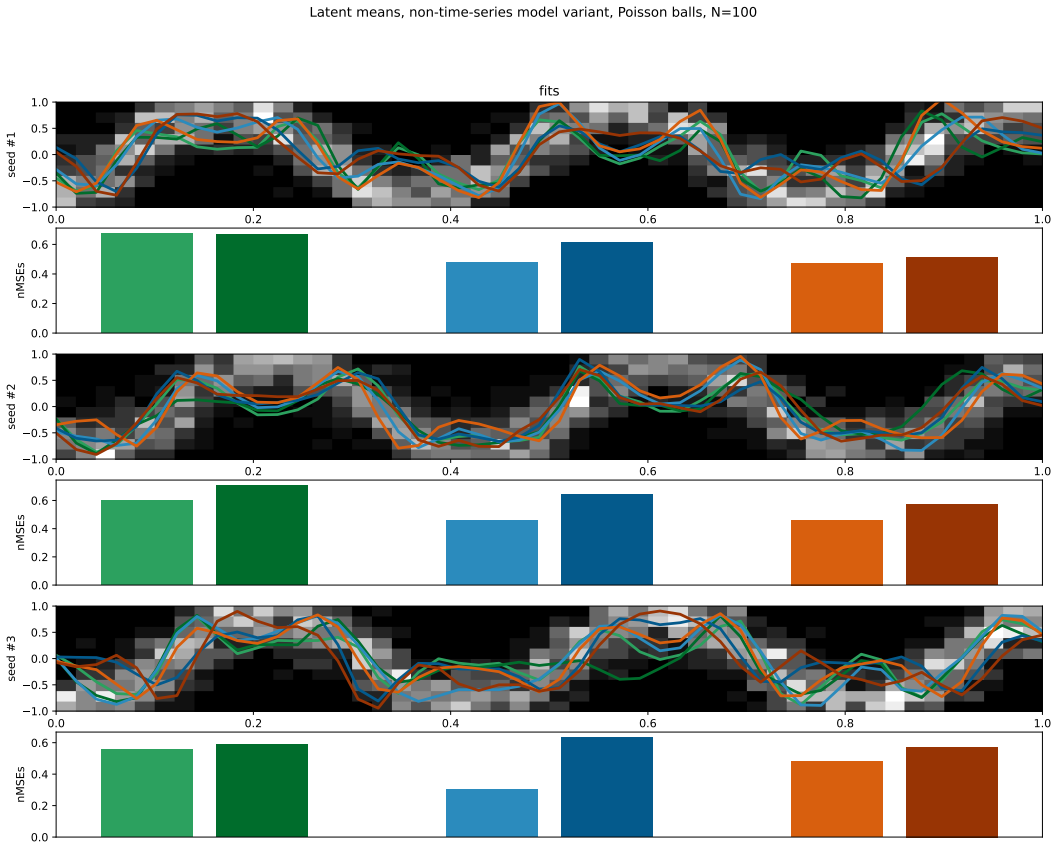

Figure A.23: Posterior means $\mathbb{E}_{q(\mathcal{Z}|\mathcal{X})}[\mathcal{Z}] = \nabla\Phi(\eta_q(\mathcal{X}))$ obtained from final fits of different RPM training frameworks for $N = 100$ Poisson ball datapoints. Colors as in Fig 2. Top rows: single training data example point overlay with posterior means, bottom rows: normalized MSE between posterior means and true latents averaged across all $N = 100$ training data points. All results are bad, though better than for $N = 10$ (cf. Fig A.20).

A.7.2 standard (non-time-series) RPM on textured balls

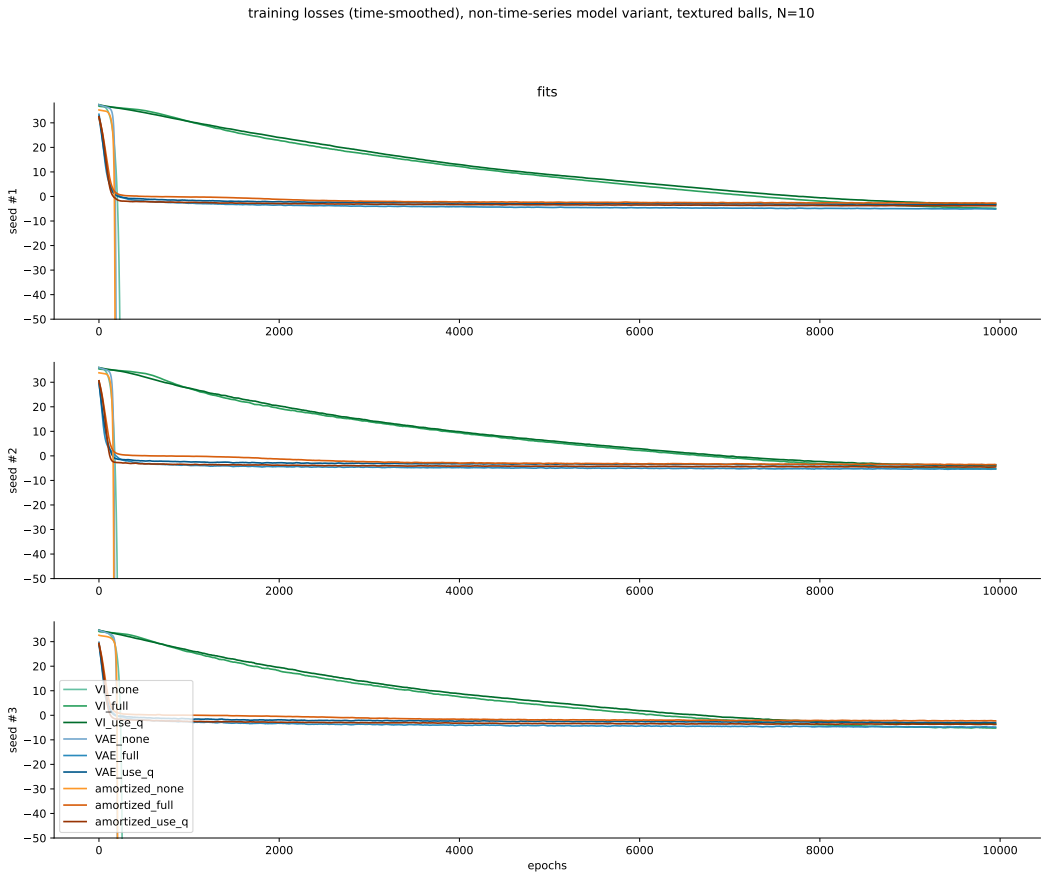

Figure A.24: training losses on $N = 10$ for textured bouncing balls, using a standard (non-time-series) RPM model. Number of epochs equals number of gradient steps (batch-size 8). All models with non-parametric natural parameters $\eta_q^{(n)}$ for the recognition model $q(\mathcal{Z}|\mathcal{X}^{(n)})$ collapsed to negative infinite loss within a few hundred gradient steps.

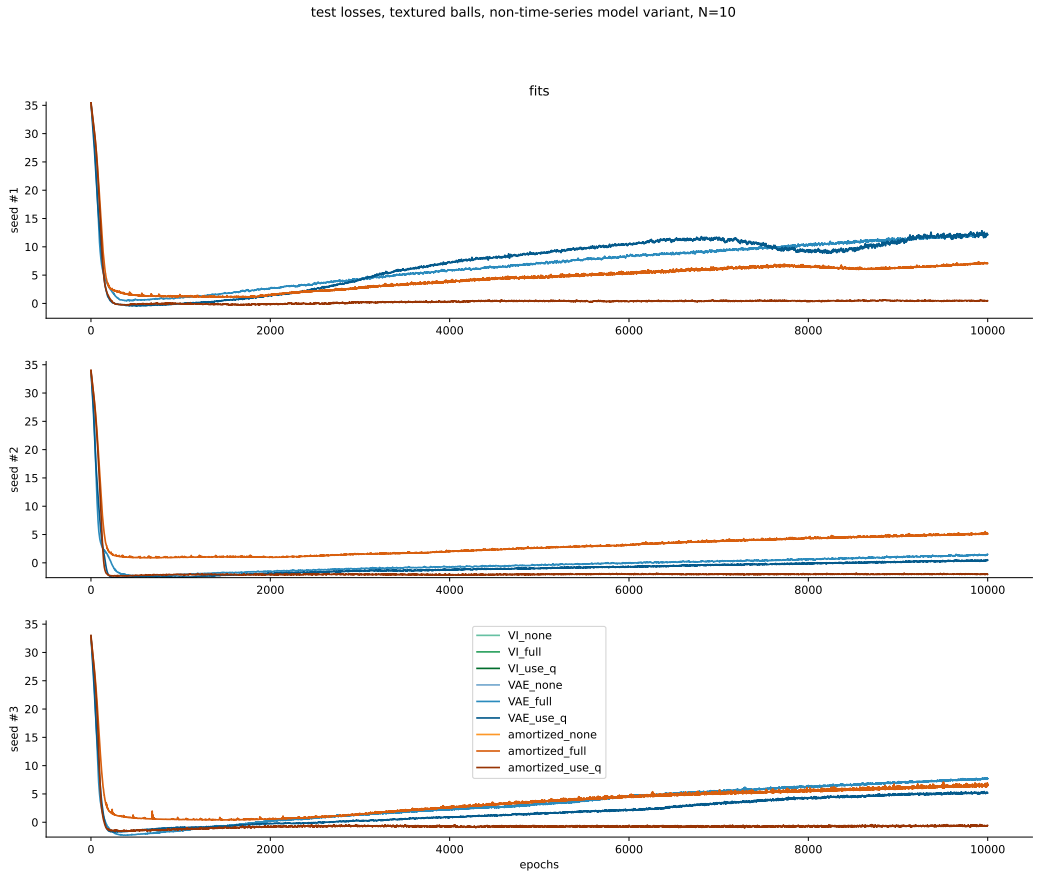

Figure A.25: test losses on $N = 10$ for textured bouncing balls for the four fully amortized RPM training frameworks, using a non-time-series RPM model. Number of epochs equals number of gradient steps (batch-size 8). We again see mild overfitting for VAE-type RPM training frameworks, in particular 'full VAE'. For these longer fits, we also see visible overfitting in 'luxury amortized' (separate neural networks for $\tilde{\eta}_j$). 'full VAE' and 'minimalist amortized' are tied for the best fits.

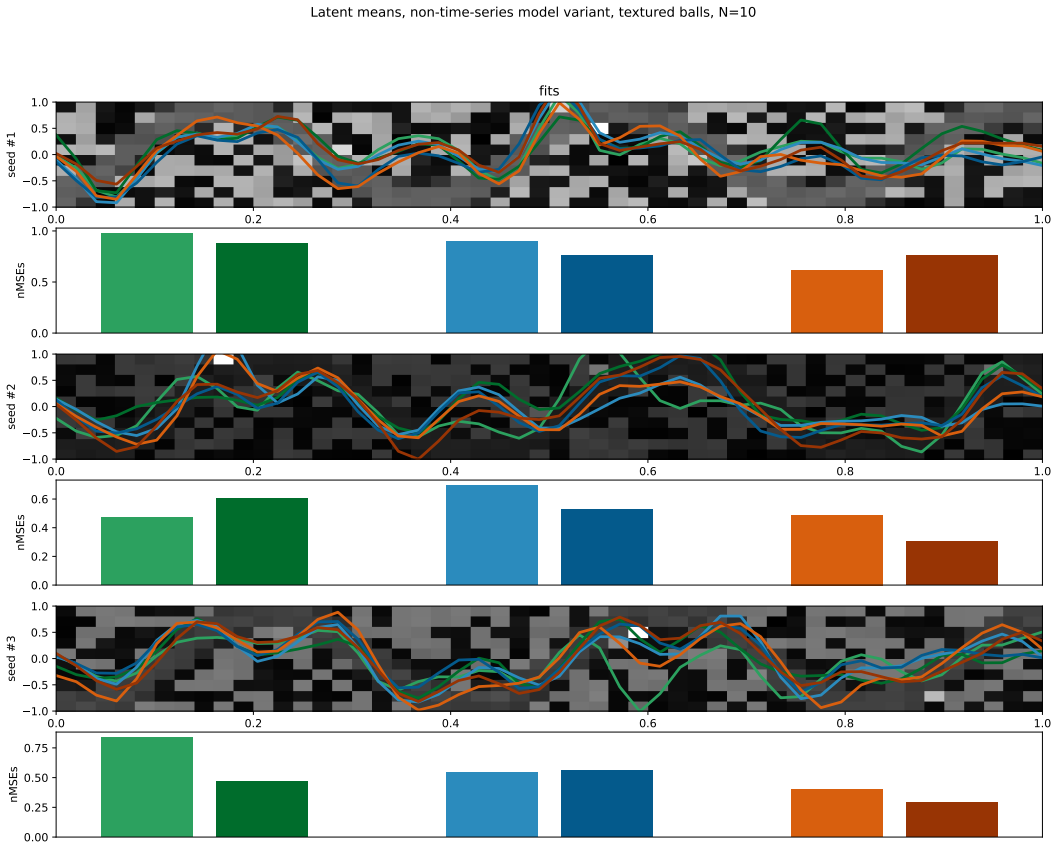

Figure A.26: Posterior means $\mathbb{E}_{q(\mathcal{Z}|\mathcal{X})}[\mathcal{Z}] = \nabla\Phi(\eta_q(\mathcal{X}))$ obtained from final fits of different RPM training frameworks for $N = 10$ textured ball datapoints, for non-time-series RPM. Colors as in Fig 2. Top rows: single training data example point overlay with posterior means, bottom rows: normalized MSE between posterior means and true latents averaged across all $N = 10$ training data points. Latents are throughout much worse than for time-series RPM, and overall 'minimalist amortized' works best. Fits with non-parametric η_q not included since they all broke (and I neglected to checkpoint the last working update step before the weights turned NaN...).

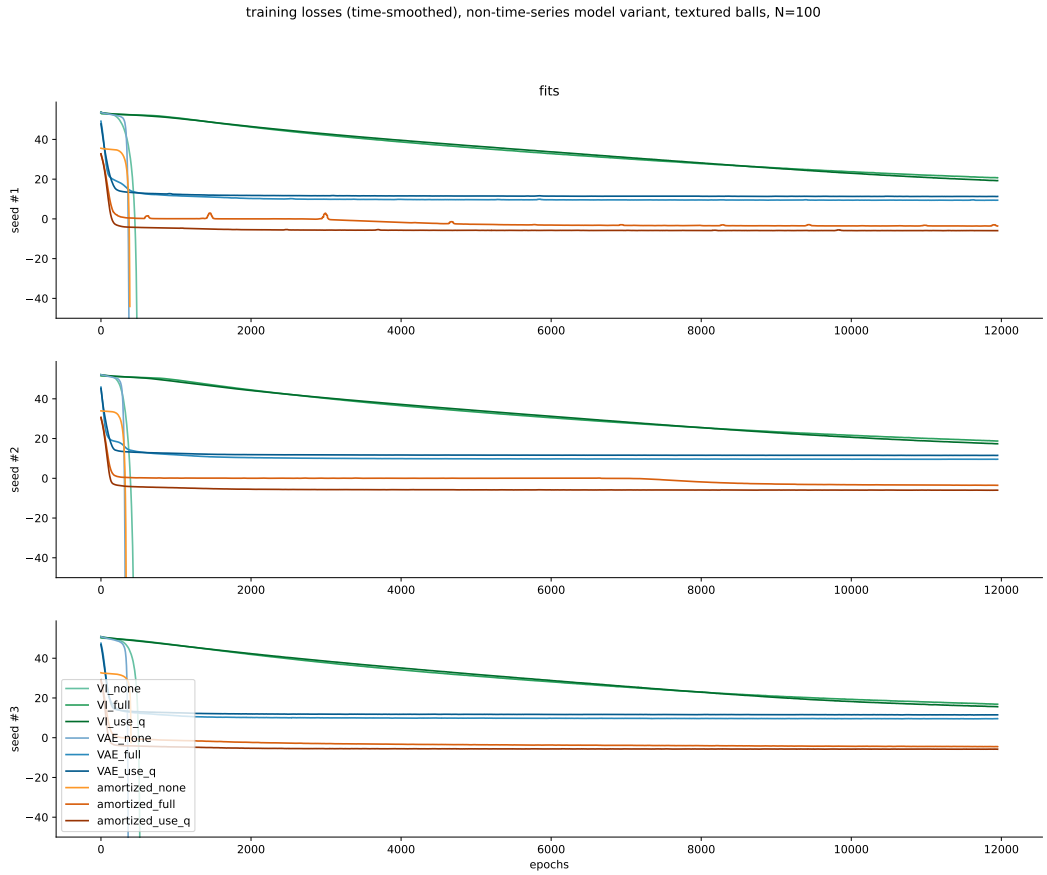

Figure A.27: training losses on $N = 100$ for textured bouncing balls, using a non-time-series RPM model. Training losses are smoothed using a running average over 50 training steps. Note how the training frameworks with non-parametric recognition parameter $\eta_q^{(n)}, \tilde{\eta}_j^{(n)}$ again diverge to negative infinite loss. Training frameworks with non-parametric $\tilde{\eta}_j^{(n)}$ but parametric $\eta_q(\mathcal{X})$ stay stable, but fail to converge within 12.000 training steps, again presumably due to the sparser training signal on the $N = 100$ training set (cf. $N = 10$ in Fig A.24).

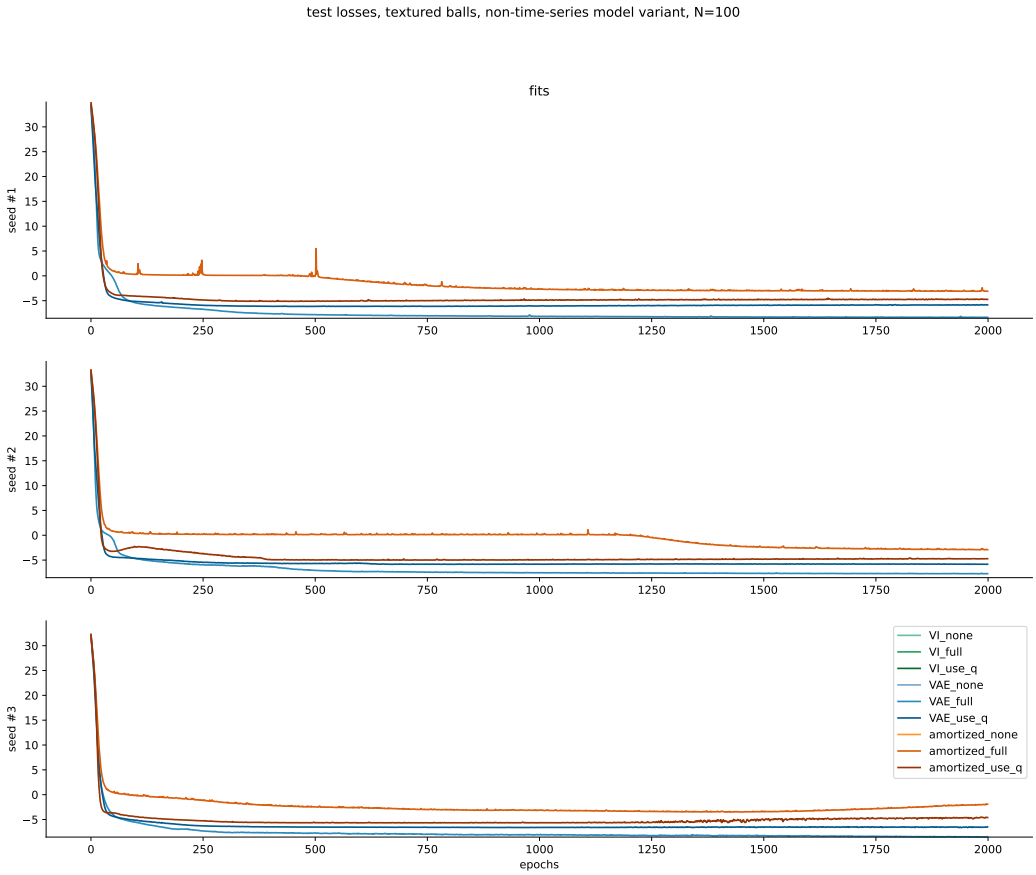

Figure A.28: test losses on a held-out test dataset set with $N_{\text{test}} = N = 100$ for textured bouncing balls for the four fully amortized RPM training frameworks, using a non-time-series RPM model, over 2000 training epochs. Note how there is no visible overfitting, but the performance ordering is still flipped ('full VAE' best) compared to the training errors in Fig. A.27 (where 'minimalist amortized' is best).

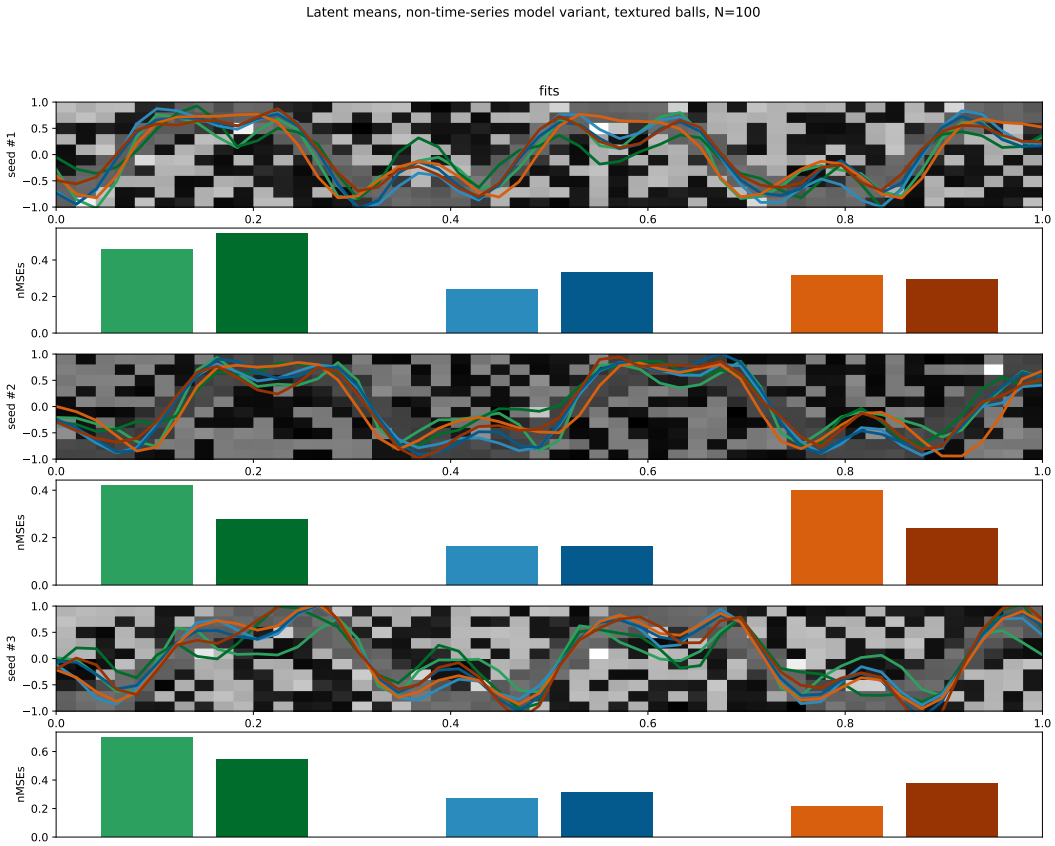

Figure A.29: Posterior means $\mathbb{E}_{q(\mathcal{Z}|\mathcal{X})}[\mathcal{Z}] = \nabla\Phi(\eta_q(\mathcal{X}))$ obtained from final fits of different RPM training frameworks for $N = 100$ textured ball datapoints. Colors as in Fig 2. Top rows: single training data example point overlay with posterior means, bottom rows: normalized MSE between posterior means and true latents averaged across all $N = 100$ training data points. All results are bad, though better than for $N = 10$ (cf. Fig A.26).