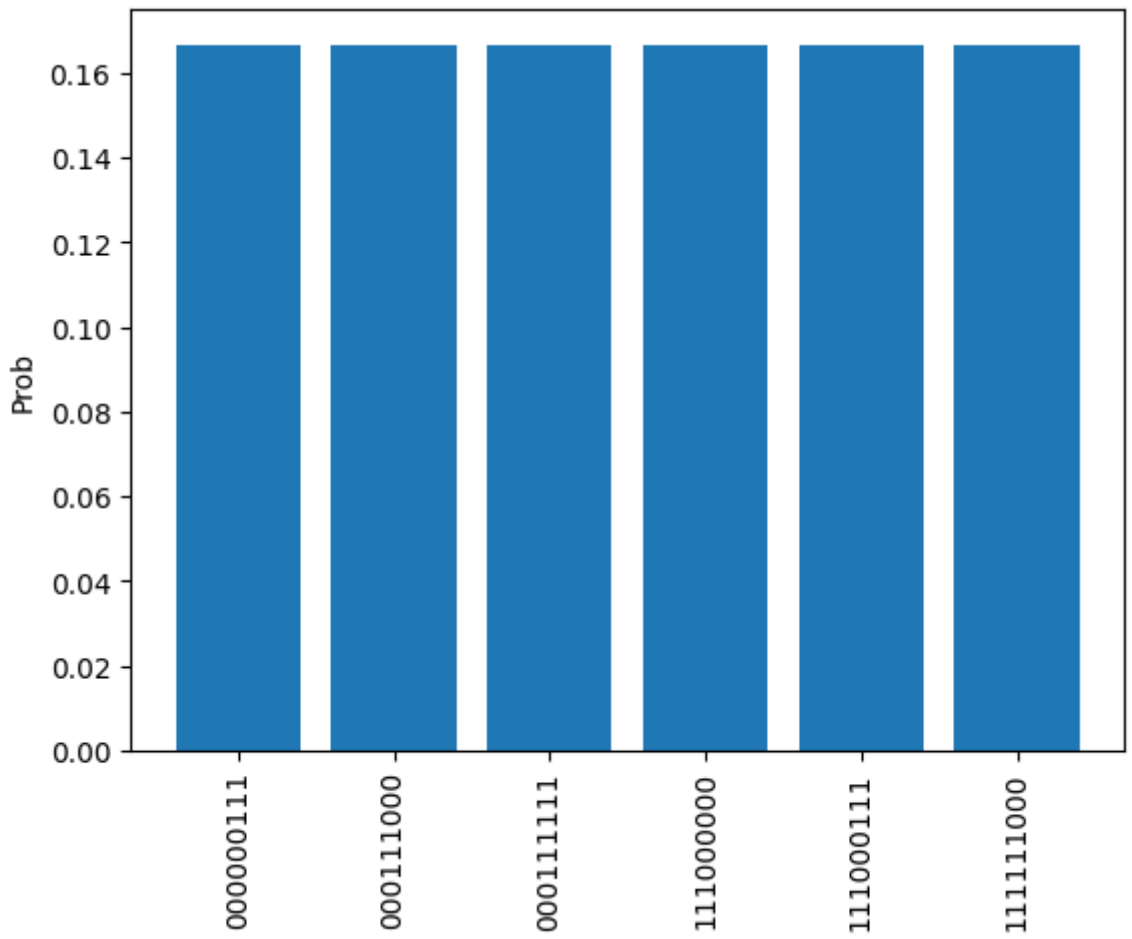
Note on MIXER CONSTRUCTION & EFFECTS

So here I will sum up the effects of mixers for the purpose of sampling (and training) from ising-type models. We have already seen the effect of using 'vanilla' mixers for sampling from

MIXER CONSTRUCTION from DATA

Observations

For the sake of simplicity, I will base my reasoning about the case of BAS (just bars for now) dataset. The dataset is given by $\mathcal{D}^{b3}$:

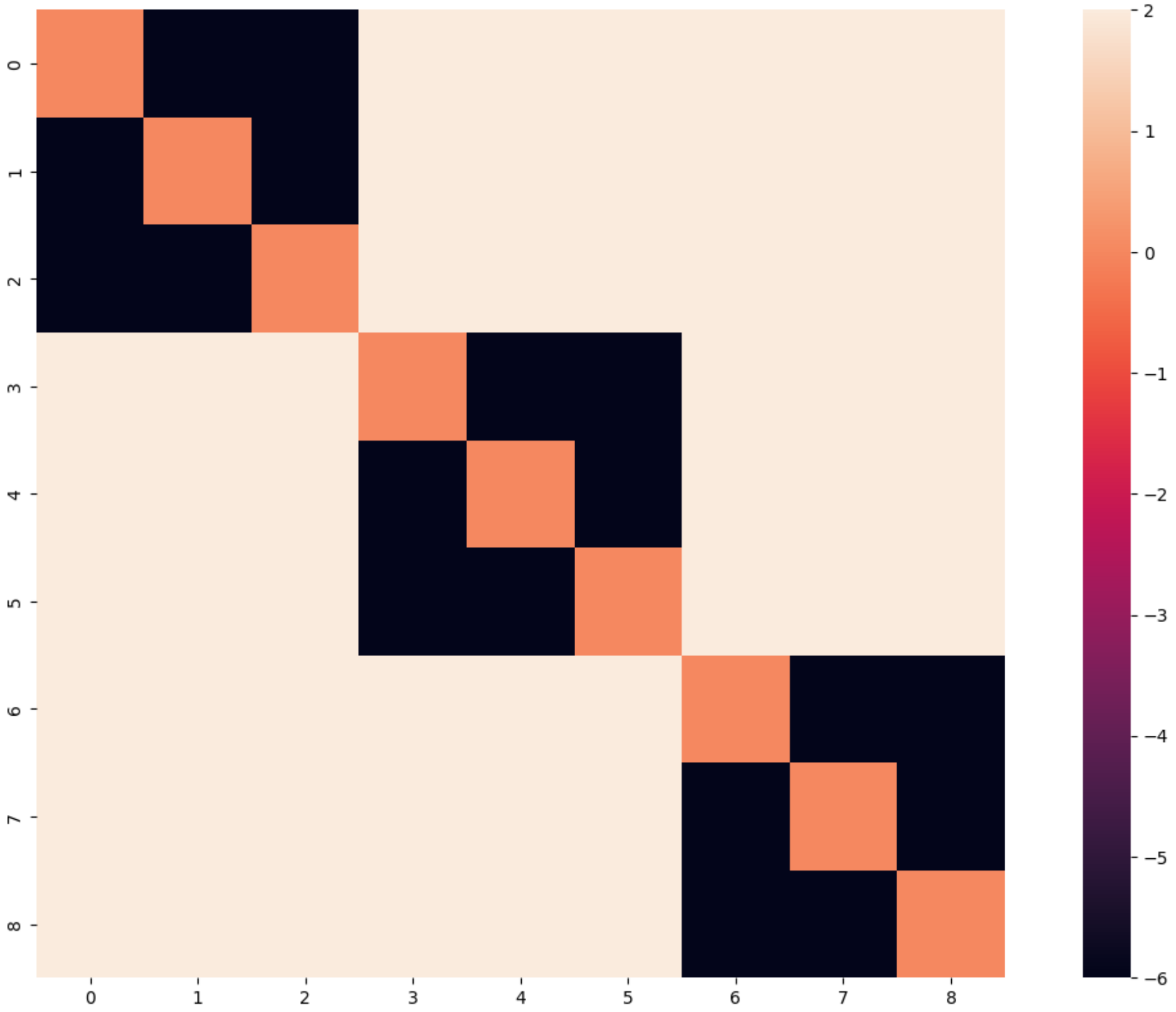


It should be noted that such distributions have equal probability distribution over all elements of the set, though its a trivial fact it might be useful for some deductions later.

In training 'Boltzmann Machines' we are aiming to find a suitable binary interactions and biases for our model ; $\mathcal{M}(J,h) = \mathcal{M}(J_{ij}, h_i)$ so that the corresponding boltzman distribution corresponds to the data set.

However for datasets of small size we can use methods like 'Hebbian Learning' to construct a model in similar fashion, which is also something we used to get a baseline model for comparing the effectivity of the sampling routines.

The model M^{b3}_{hebb} constructed is depicted here:



The non-diagonal terms represent the interaction strengths (J_{ij}) and the terms in the diagonal indicates the biases (h_i) which when expanded mathematically have the structure

$$M^{b3}_{hebb} = (-6)\left((Z_0Z_1 + Z_1Z_2 + Z_0Z_2) + (Z_3Z_4 + Z_4Z_5 + Z_5Z_3) + (Z_6Z_7 + Z_7Z_8 + Z_8Z_6)\right) \; + \; 2\,(..\text{rest of the interactions}..) \; + \; 0\,(\text{biases})$$

We can draw a close analogy with *Stabilizer Hamiltonians*, i.e hamiltonians which have as their ground states the positive eigenspace of the elements in *Stabilizer Group*.
Its easy to recognise that the pauli terms with -6 as their coefficient are all part of the stabilizer group of the well known *3-qubit repetition QEC* concatenated 3 times.

Ideally the stabilizer hamiltonian should be constructed as the negative sum of all the generators of the stabilizer group i.e $\mathcal{S}^{b3} =< Z_0Z_1, Z_1Z_2, Z_3Z_4, Z_4Z_5, Z_6Z_7, Z_7Z_8 >$, thus the hamiltonian is

$$M^{b3}_{stab} \; = \; -(Z_0Z_1 + Z_1Z_2 + Z_3Z_4 + Z_4Z_5 + Z_6Z_7 + Z_7Z_8)$$

We can verify that both M^{b3}_{hebb} , M^{b3}_{stab} have the same ground states (except states with all zeros or all ones), however they have different i energy level distribution of excited levels. The additional terms (with coefficient +2) in M^{b3}_{hebb} have the effect to introduce larger energy gap between the ground-states and non ground states of the system, which in turn makes it easier for sampling algorithms to distinguish between them.

Coming to better mixer constructions, ideally we would like to have mixer hamiltonians T which only allow for transitions between the elements of the dataset (or more generally elements that have a significantly higher probability than the rest).

Though under general circumstances it is hard to create such operators, we can use some ideas from *stablizer codes* for cases where the dataset can be related to ground states of a stabiliser hamiltonian. In QEC any non-trivial operator that maps a stabilized state to another stabilized state is simply a 'logical operator', it has the property that it commutes with the elements in the stabilizer group.

Thus for the elements of $\mathcal{D}^{b3}$, the ideal mixer would be the one generated by $\mathcal{L}^{b3}_X =< X_0X_1X_2, X_3X_4X_5, X_6X_7X_8 >$, which are simply the 'logical X' of the *3-qubit repetition QEC*. As expected the operators in $\mathcal{L}^{b3}_X$ commute with those in $\mathcal{S}^{b3}$ and thus with the corresponding hamiltonian M^{b3}_{stab}.
However, note that the mixer terms doesn't commute with M^{b3}_{hebb} , because of the *non-stabilizer* terms in the hamiltonian.

It should be noted that in generic QAOA type circuits are built with interspersed layers of Problem (P) and Mixer Hamiltonian (T) that do not mutually commute with each other, $[T, P] \neq 0$.

The main purpose of having non-commuting layers is to facilitate transitions between states with different energies, thus allowing for the algorithm to get an overview of the global energy landscape of the system which is necessary for optimisation / sampling tasks.

Note that in case $[T, P] = 0$, the corresponding time-evolution unitary would fail to facilitate transitions between states of different energy levels. Usually such choice of mixer hamiltonians would be disadvantageous, except if all the states of interest have the same energy, which is the case here with hamiltonians like M^{b3}_{stab} , M^{b3}_{hebb} . Thus constructing T out of the elements in $\mathcal{L}^{b3}_{X}$ would be sufficient to induce necessary transitions.

Generalization

These ideas could be generalised as follows. Say we are interested in a classical sampling / optimisation process where the problem hamiltonian is P, which is by requirement a diagonal hamiltonian (i.e composed of only Pauli Z terms) but we are interested in just the states (computational basis states) within a dataset $\mathcal{D}$ ($\mathcal{D} \subset \{1, 0\}^n$).
We can assume there exists a an oracle $O_{\mathcal{D}}$, with the property

$$O_D|\psi\rangle = \begin{cases} |\psi\rangle \ ; \ |\psi\rangle \in \mathcal{D} \\ \gamma_\psi \, |\psi\rangle \ ; \ |\psi\rangle \notin \mathcal{D} \ , \gamma_\psi \neq 1, \end{cases}$$

that the elements from the dataset are in +1 eigenspace of the O_D. The fact that such an operator would exist and would be a diagonal unitary is easy to realise.
Given this, we can say that the ideal choice of the mixer operator would be T, such that

$$[T, O_{\mathcal{D}}] = 0$$

which makes sure that all transitions induced by T preserves the dataset $\mathcal{D}$. But note also that under general circumstances we would also like to have

$$[T, P] \neq 0$$

to ensure that T is able to induce transitions between different energy levels. Of course this would not be a requirement anymore if all states in $\mathcal{D}$ have the same energy, as in the case with stabilizer hamiltonians.

For example with $\mathcal{D}^{b3}$, we can pick the oracle to be

$$O_{\mathcal{D}^{b3}} \ = \ \left(\frac{-1}{6}\right) M^{b3}_{stab}$$

and $T = \mathcal{L}^{b3}_{X} =< X_0X_1X_2, X_3X_4X_5, X_6X_7X_8 >$, as for the the problem hamiltonian P we can either set

- $P = M^{b3}_{stab}$; Then we will have $[T, P] = 0$, since all elements of the dataset have same probability it shouldn't be a big issue.
- $P = M^{b3}_{hebb}$; The presence of non-stabilizer terms (like 2 $(Z_0Z_3 + Z_1Z_7 + \ldots)$ will lead to $[T, P] \neq 0$ since this terms would anti-commute. Since the elements of the dataset still have same energy we do not gain in the criterion, however the sampling might benefit from the larger energy gap between non-ground states.

Speaking of generality, we should note that for most practical datasets it is might be highly troublesome to find suitable operators for T and O_D. We have been able to deal with our example only because its related to an instance of stabilizer-hamiltonian, but in general case it might take exponential amount of computation.

EFFECTS of DATA BASED MIXERS

So next we can study the effects of using mixers constructed out of the properties of Datasets, for problems like sampling and training boltzman machines. The example are all based on the experiments run on the bars dataset $\mathcal{D}^{b3}$.

For the purpose of simulation we have used the M^{b3}_{hebb} as our problem hamiltonian. It was indicated in the previous section that this hamiltonian has the same ground states as M^{b3}_{stab} but had larger energy difference between ground-states and non ground states.

Following the discussion in the last section we know how to construct ideal mixer that would restrict transitions to those that preserve the Dataset constraint. For the sake of comparison we will also consider mixers that do not preserve the Dataset, since we don for example

- Mixers hamiltonian of weight 2, T_{w_2} : Operators of form X_iX_j
- Mixers hamiltonian of wight 3, T_{w_3} : Operators of form $X_iX_jX_k$
  and so on. Since we do not have any specific choice over the configuration (i.e choice of the indices $i, j, k. .$) of those operators, we construct the final mixer by summing over all possible configurations, i.e

$$T_{w_3} = \sum_{\substack{i,j,k=1 \\ i \neq j \neq k}}^{n} X_iX_kX_k$$

  thus in general any mixer of weight w, T_w is a sum over $\binom{n}{w}$ pauli strings of weight w.

Though such mixers do not satisfy $[T, O_D] = 0$, we will see from our experiments that they might be more helpful for practical purposes.

NB : To denote the type of MCMC used we will stick to the following convention:
mcmc type -> `<cl/qu>-<mixer type and weight>`
Eg. `qu-wt1` -> quantum MCMC where the mixer is T_{w_1} ;
`qu-stabilizers-wt3` -> quantum MCMC where the mixers are based on the stabilizer hamiltonians and are of weight 3; i.e the mixer hamiltonian is

$T_{D^{w3}} = X_0X_1X_2 + X_3X_4X_5 + X_6X_7X_8$

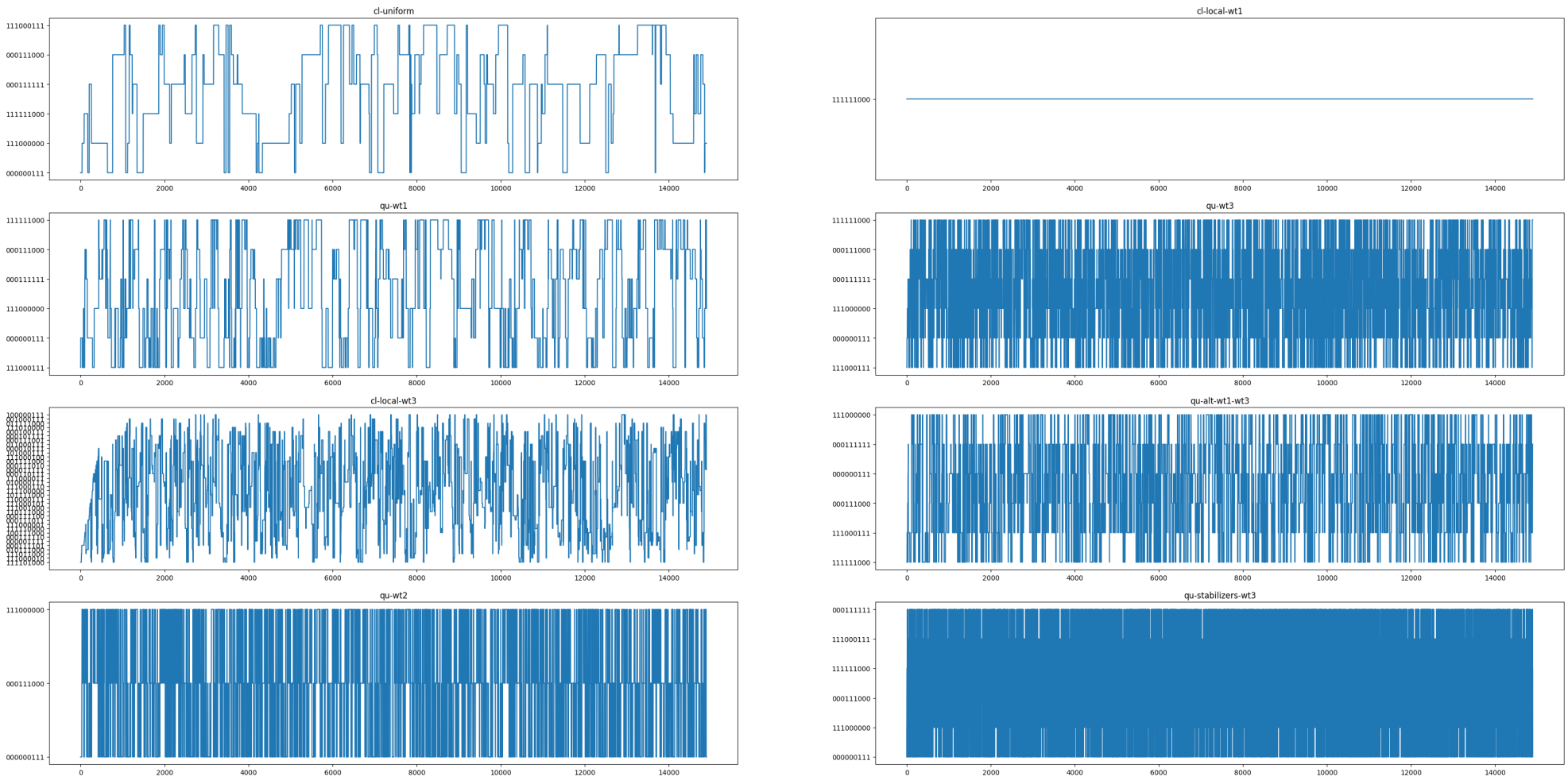
`cl-local-wt3` -> classical MCMC, where only states that are $3h_D$ are proposed

Sampling

Here we run 10 independent instances for each of the MCMC types : `['cl-uniform', 'cl-local-wt1', 'qu-wt1', 'qu-wt3', 'cl-local-wt3', 'qu-alt-wt1-wt3', 'qu-wt2', 'qu-stabilizers-wt3']`

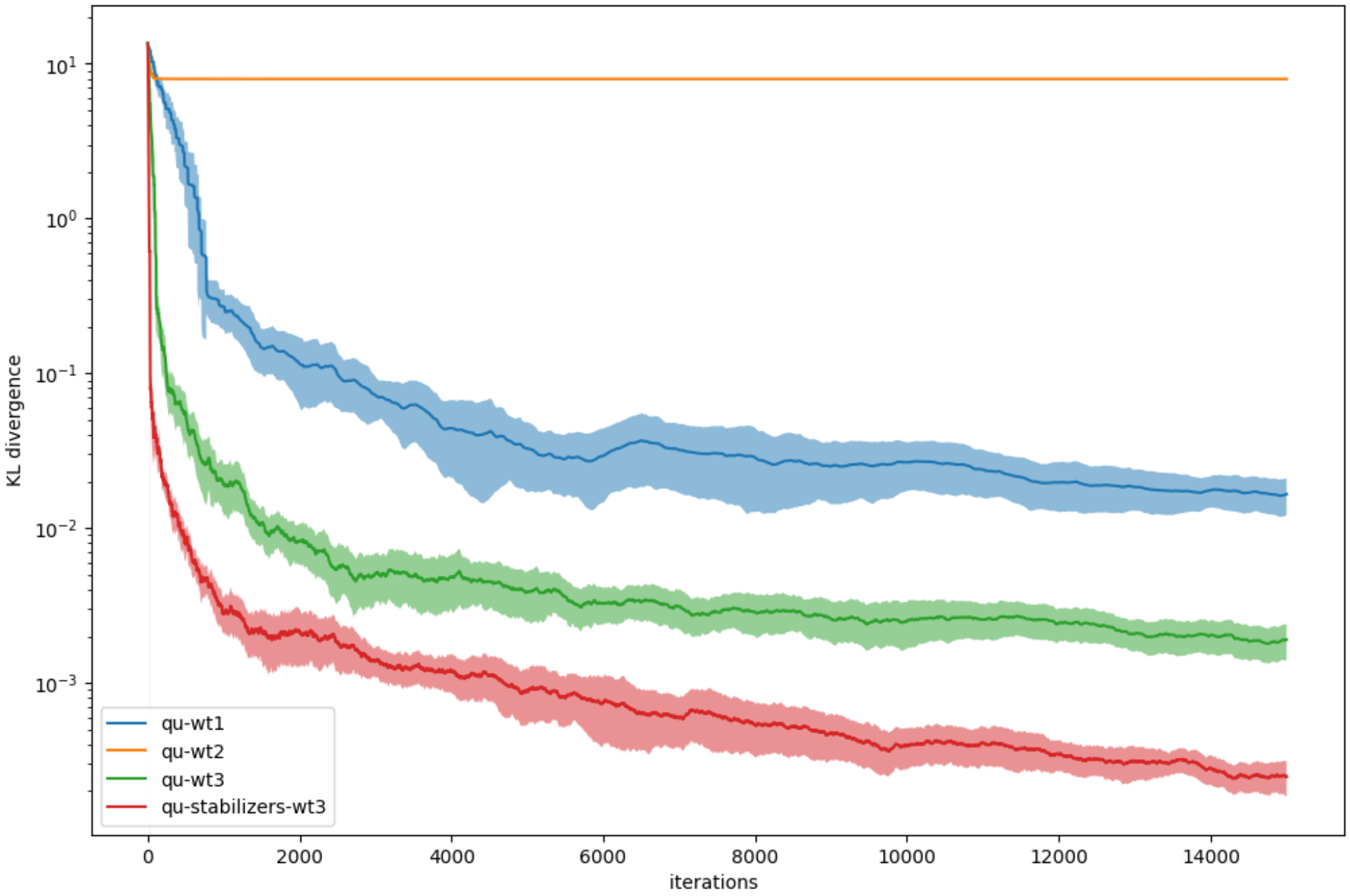
NB. : For all experiments we initiate the MCMC chain from an element of the dataset. It will be clear later that the former is a stringent requirement for some mixers to perform sampling.

Trajectory Visualisation



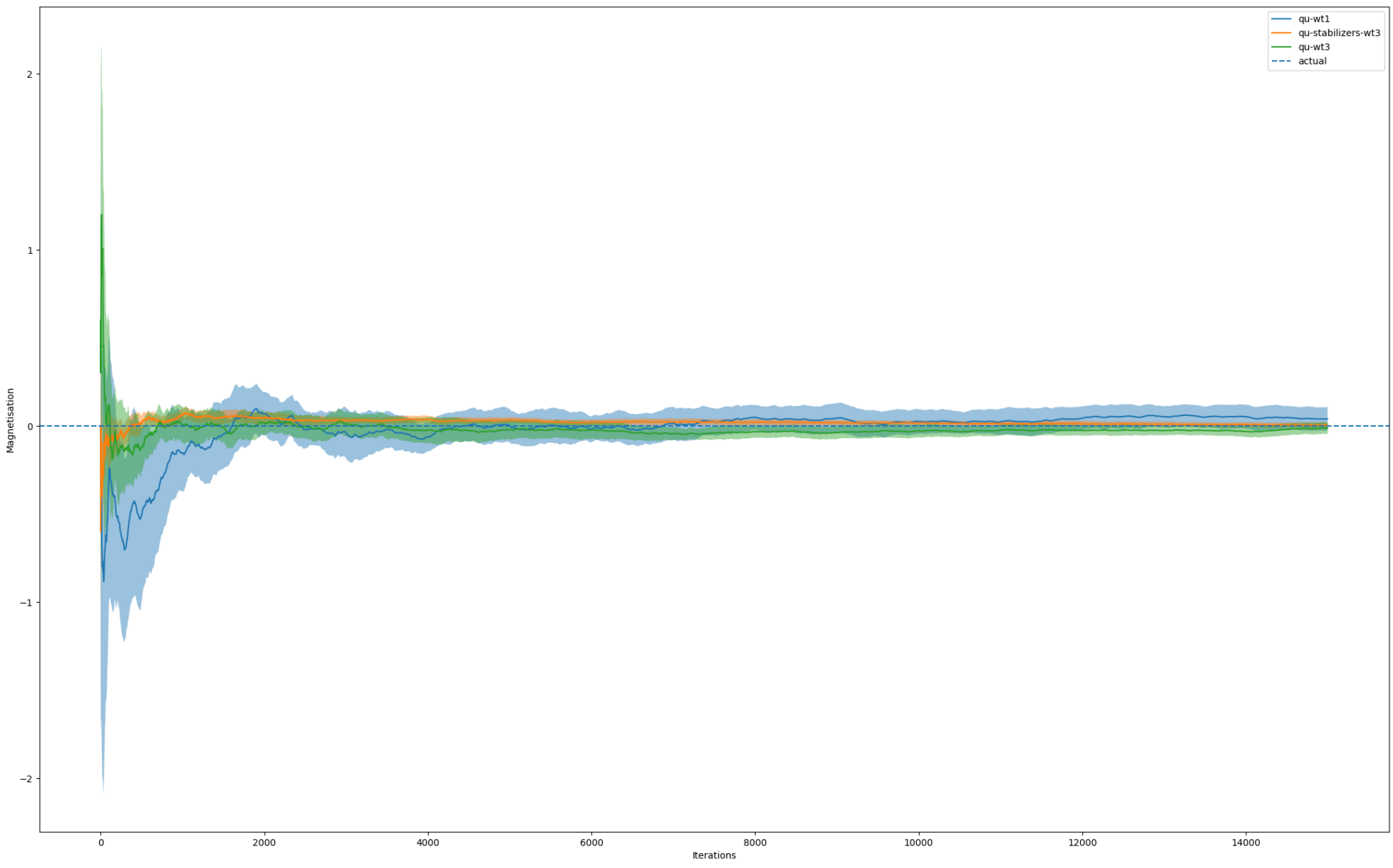
Despite the differences, most mcmc types manage to sample the all the ground states. Except `cl-local-wt1` which fails to make any transitions and `qu-wt2` which stays transitioning between just 3 states. All except the mcmc chain for `cl-local-wt3` remains confined to the elements of the dataset $\mathcal{D}^{b3}$

KL Divergence

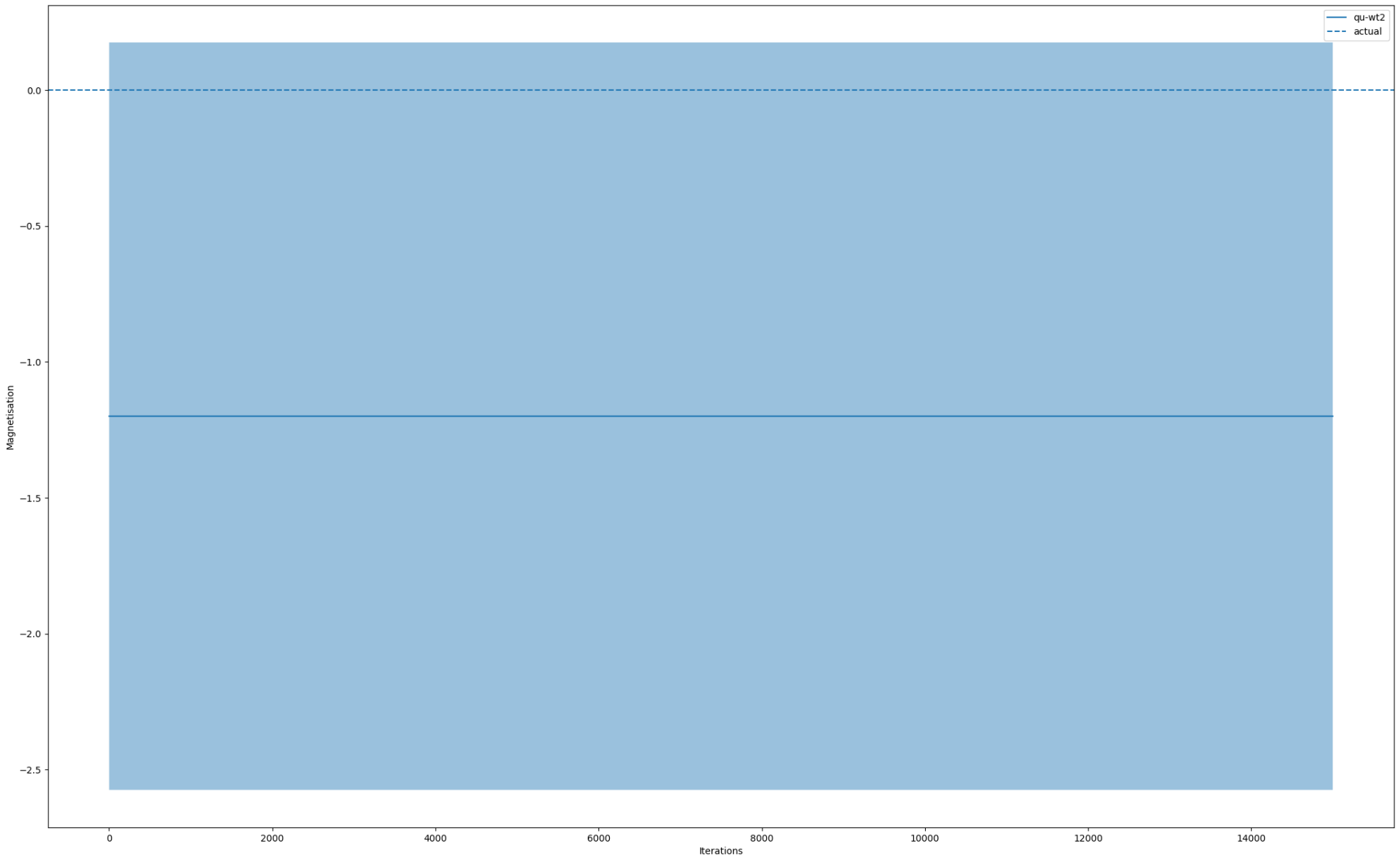


Here `qu-stabilizers-wt3` is clearly the winner. It also shows that `qu-wt2` that fails to sample the hamiltonian even to the worst accuracy, this also apparent from the last plot, nonetheless we have seen that it preserves the dataset constraint.

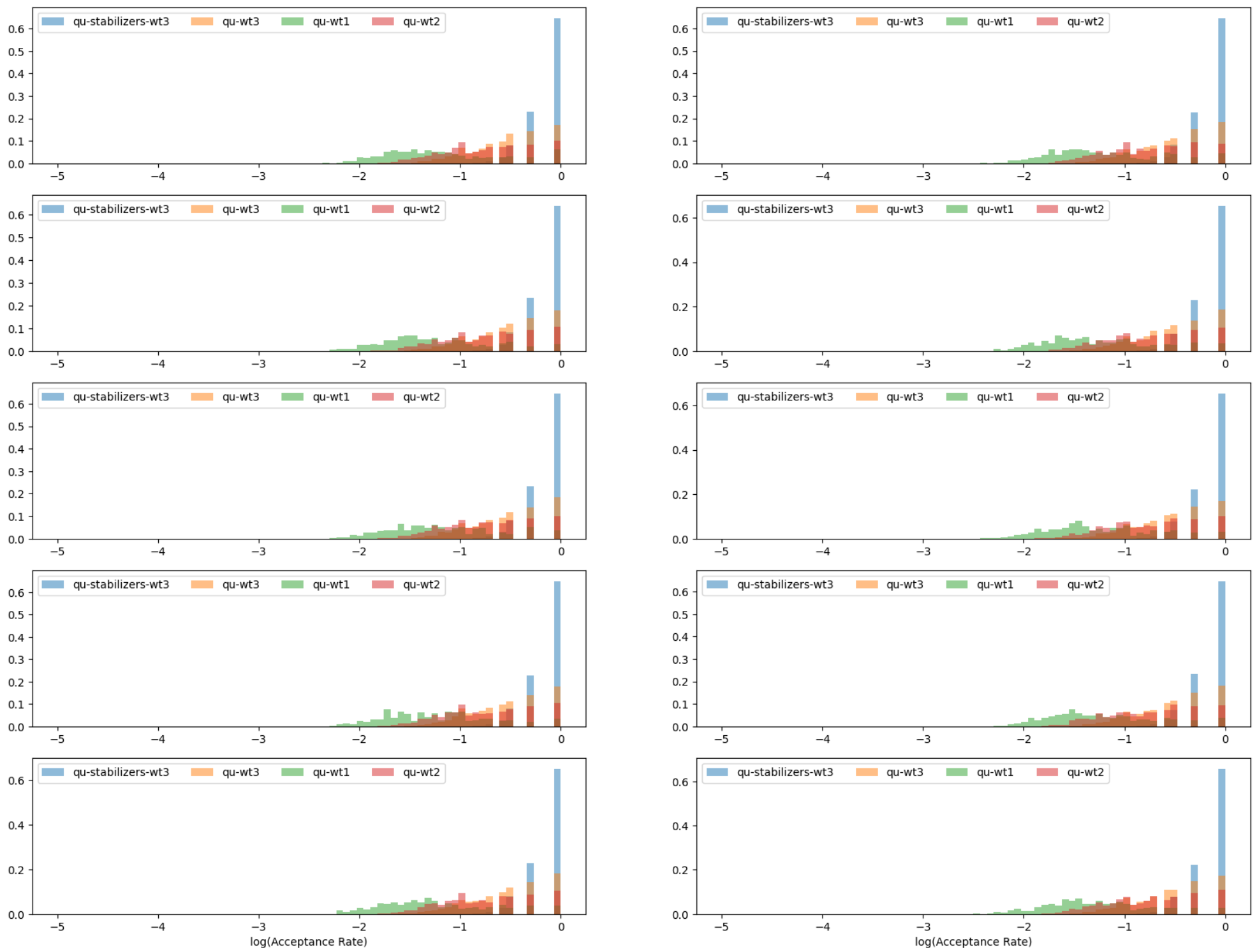
Magnetisation

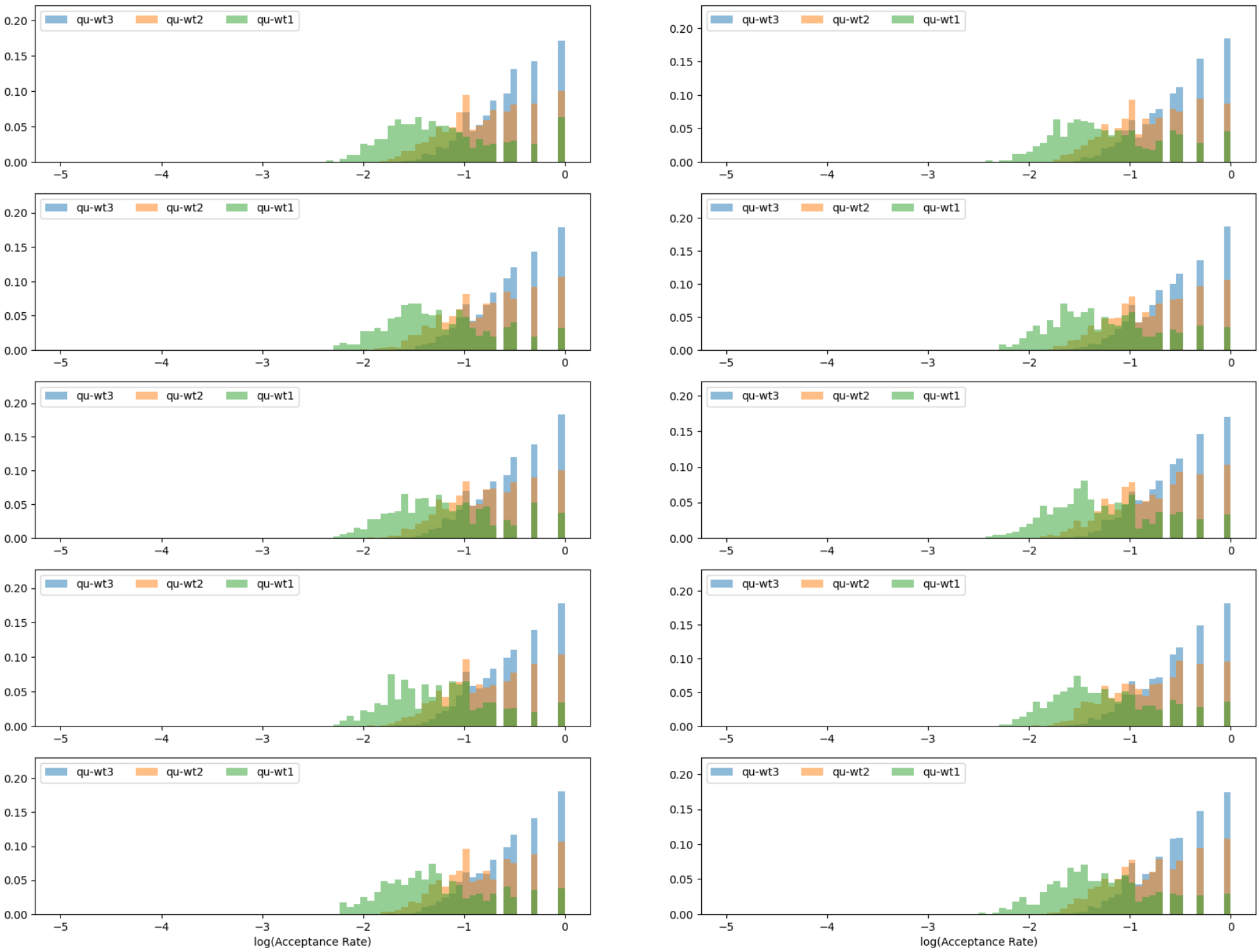


The performance in this case as similar to the previous one. The case for `qu-wt2` fails disastrously as expected, and doesn't even approach the actual magnetisation.



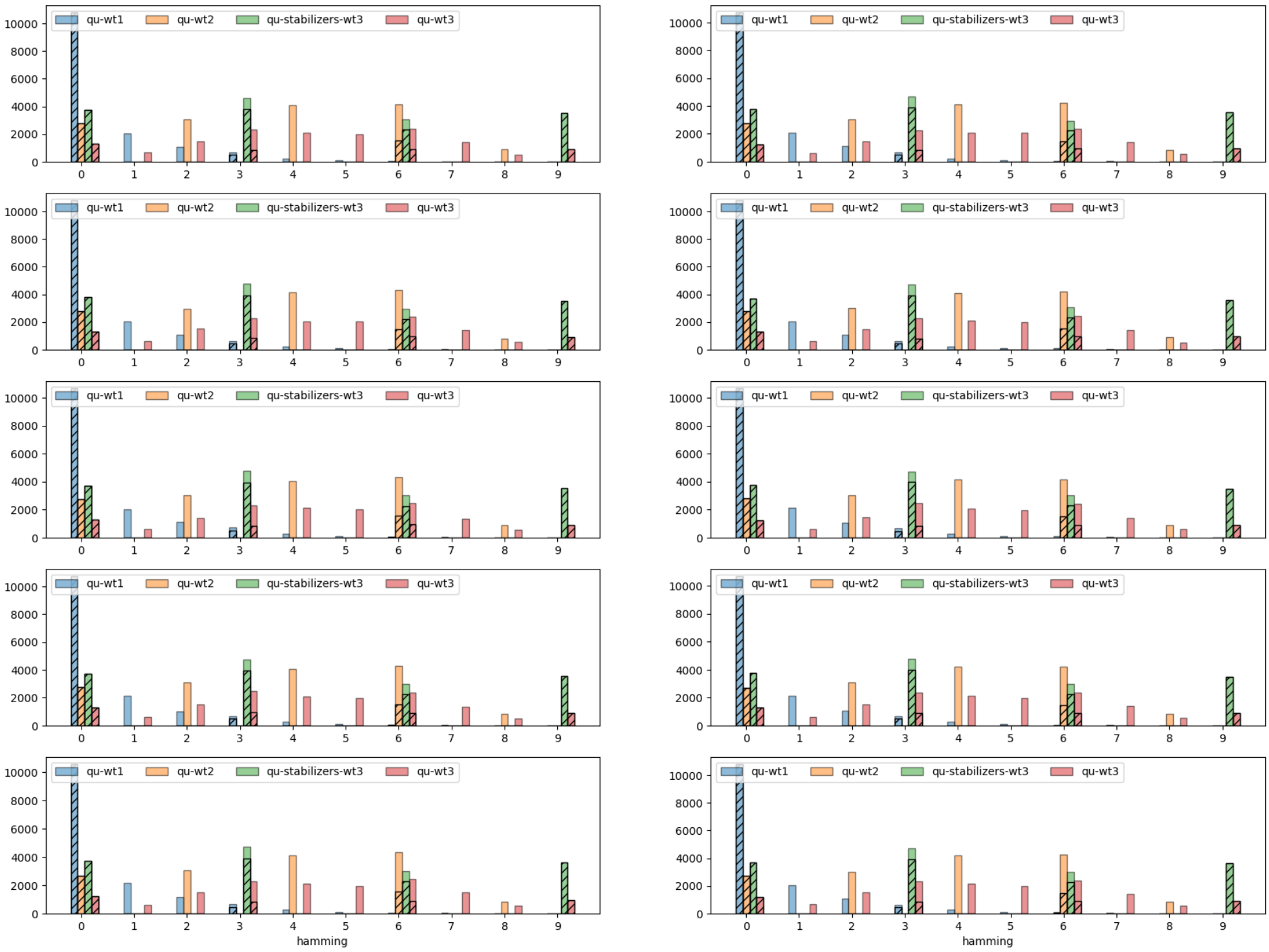
Acceptance Probability





- `qu-stabilizers-wt3` has the best acceptability as expected.
- The distribution of `qu-wt2` is rather surprising, and highlights one of the key properties about customised mixer hamiltonians.
  - The plot shows that propositions made by `qu-wt2` are well accepted but yet we know from the KL-Div plot that it barely manages to sample the distribution,
  - This happens because the mixer in `qu-wt2` restricts the propositions made by the sampler to a few elements within the data set which are likely to be accepted but fails to propose most elements in the dataset, thus reducing its effectivity as a sampler.

Hamming Distance



NB. : The shaded portions indicate the fraction of the propositions that were accepted

Here we see that

- That only transitions of h_d in multiple of 3 are accepted in significant proportions.
- For `qu-stabilzers-wt3` ; the proposed transitions are restricted to states differing by multiples of $3\ h_d$, and most of the m are accepted.
- For `qu-wt2` ; the proposed transitions are restricted to states differing by multiple of $2\ h_d$, but only propositions of $6\ h_d$ are accepted.
- For `qu-wt3` ; the proposed transitions are spread over all h_d

To some extent, I think the data can be explained by observing the terms in the mixer unitary (i.e unitary corresponding to mixer hamiltonian).

- For `qu-stabilizers-wt3` :

  $$exp(i\gamma T_{D^{w3}}) = (c_\gamma\ I + is_\gamma\ X_0X_1X_2)\ (c_\gamma\ I + is_\gamma\ X_3X_4X_5)\ (c_\gamma\ I + is_\gamma\ X_6X_7X_8)$$

  where $c_\gamma = cos(\gamma)$; $s_\gamma = sin(\gamma)$ (for convenience we use γ to represent γt_{ev} in the theory).

  The expansion will only have pauli strings of weights 0,3,6,9 because none of the three terms overlap against each other. Of course the relative weight of the pauli terms would depend on the coefficient, but it still explains our previous observation.
- For `qu-wt1` :

  $$exp(i\gamma T_{w_1}) = \prod_i\ (c_\gamma I + s_\gamma X_i)$$

  Which upon expansion will lead to operators of all possible weights, with an operator of weight w having a coefficient proportional to s_γ^w. Since we have chosen a small gamma, operators of higher weights will have lower coefficient and thus a lower effect in inducing corresponding transitions
- For `qu-wt2` :

  $$exp(i\gamma T_{w_2}) = \prod_{\substack{i,j \\ i\neq j}}\ (c_\gamma I + s_\gamma X_iX_j)$$

  Unlike the previous case here the individual terms do overlap, since the pauli strings are comprised of two elements there could be overlap on either on
  - a single element ; like $X_iX_j \cdot X_jX_k = X_iX_k$
  - or both the elements ; in which case the resultant operator is identity I . (This case is unlikely since every pair appears just once)
    But in both the cases the resultant operator has an even weight of the pauli string, this explains why every transition induced by the mixer

were of $2\ h_d$. Incase there are no overlaps it follows trivially that the strings will be of weight in multiples of $2h_d$.

- For `qu-wt3`:

$$exp(i\gamma T_{w_3}) = \prod_{\substack{i,j,k \\ i\neq j\neq k}} (c_\gamma I + s_\gamma X_i X_j X_k)$$

  Here the overlaps could be of either 2 or 1 element which creates to an overall operator of weight 2 and 4, and without overlaps it creates operators of weight 6, 9, etc. which will then interact with each other. Moreover since operators of higher weights can be created via the interaction of few terms, there relative strength doesn't attenuate as in the case of `qu-wt1`, atleast for 'not-to-bad' choices of γ.

The fact that most of the accepted states are of $h_d = 3, 6, 9$ reflects the effect of the classical-loop that accept/rejects a transitions rather than effect of the mixer that induces the transition.

Now the major aim here is certainly to design mixer such that proposed transitions are accepted with higher probability, and to that end it is apparent that `qu-stabilizers-wt3` outperforms rest of the schemes. But one should realise that we have only been able to design such a mixer using explicit information about the design, to the extent that the sampling itself doesn't add much practical value to it. Moreover, we will see in the Boltzmann Training section that using such well-designed mixers would have rather adverse effects on the training process

Training Boltzmann Machines

From the previous section its clear that picking the right mixer could significantly increase the efficiency of the MCMC sampling algorithm. Here we would like to see if we can see similar advantage in training boltzman machines by using customised mixer based MCMC for the gradient calculation process.

Intuitively we know that the efficiency of training algorithms is based on calculating the gradient of the loss function against the parameters. Here the gradient is of the form

$$\nabla_\theta \mathcal{L} = \langle O \rangle_{data} - \langle O \rangle_{model}$$

where $\langle O \rangle_{model}$ is obtained by sampling the observable $\mathcal{O}$ against using the MCMC algorithm. So its reasonable to expect the training to benefit from the effective computation of $\langle O \rangle_{model}$, and a fast converging MCMC algorithm is supposed to benefit computation of $\langle O \rangle_{model}$.

Experimental Setup

So here we have considered training a Boltzmann Machine (BM) so that it learns to emulate the bars3 dataset $\mathcal{D}^{b3}$.

We initiate the training process with the dataset $\mathcal{D}^{b3}$, a random instance of the ising hamiltonian as the model M_θ, and instructions for type of mixer to be used in the sampling process.

In our experiments we train for 300 epochs, for each of the following MCMC types: `['cl-uniform', 'cl-local-wt1', 'qu-wt1', 'qu-wt3', 'qu-wt2', 'qu-stabilizers-wt3' ]`

Running an epoch comprises of

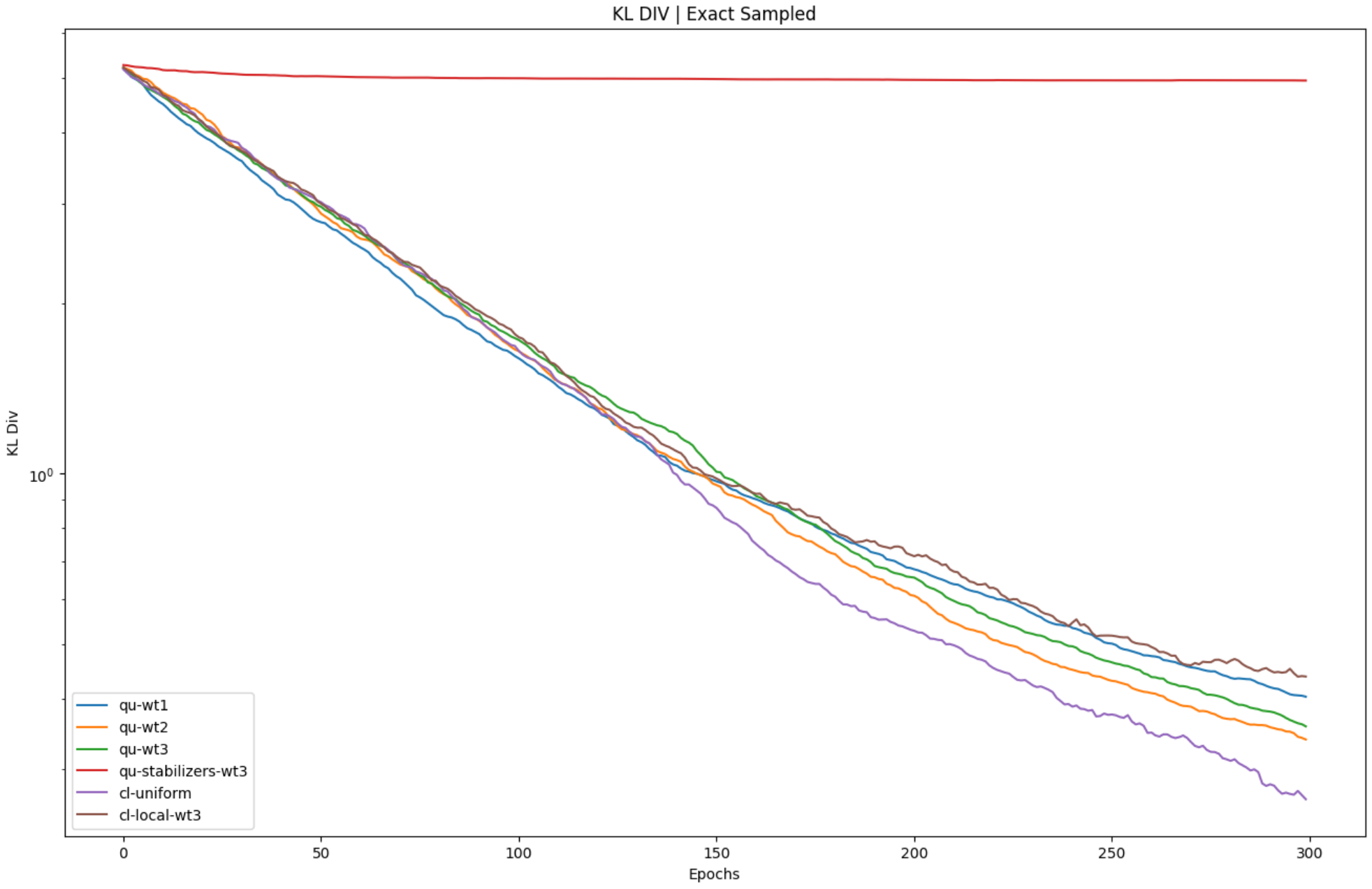
- Get markov chain by sampling the model hamiltonian M_θ for 1000 steps using the preferred choice of mixer T, as per convention we initiate the MCMC from an element of the dataset.
- Updating either all or some randomly picked of parameters (J_{ij}, h_i) of the model by calculating the gradient wrt to the markov chain obtained in the last step.

After each epoch, we calculate the KL divergence of the updated model and some statistics over the gradients calculated in that epoch (max, min, mean), and store them in the training history.

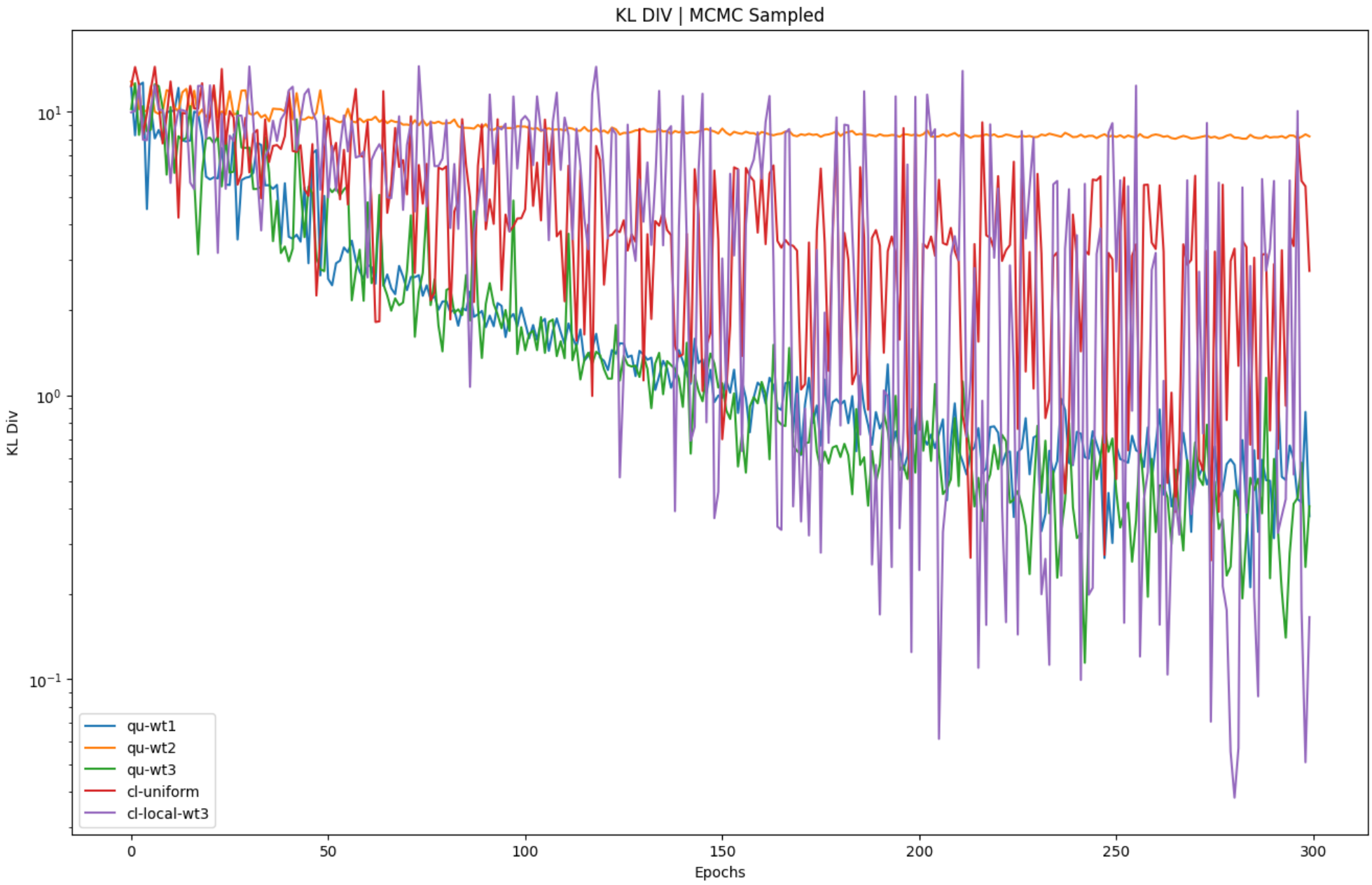
NB. : For the KL Divergence calculating step, we can either estimate it against the model distribution obtained by 'Exact Sampling' or 'MCMC Sampling' . It came to my attention that choosing it a carefully has significant effect on the KL Div of the process, for reasons that I will clarify in a while.

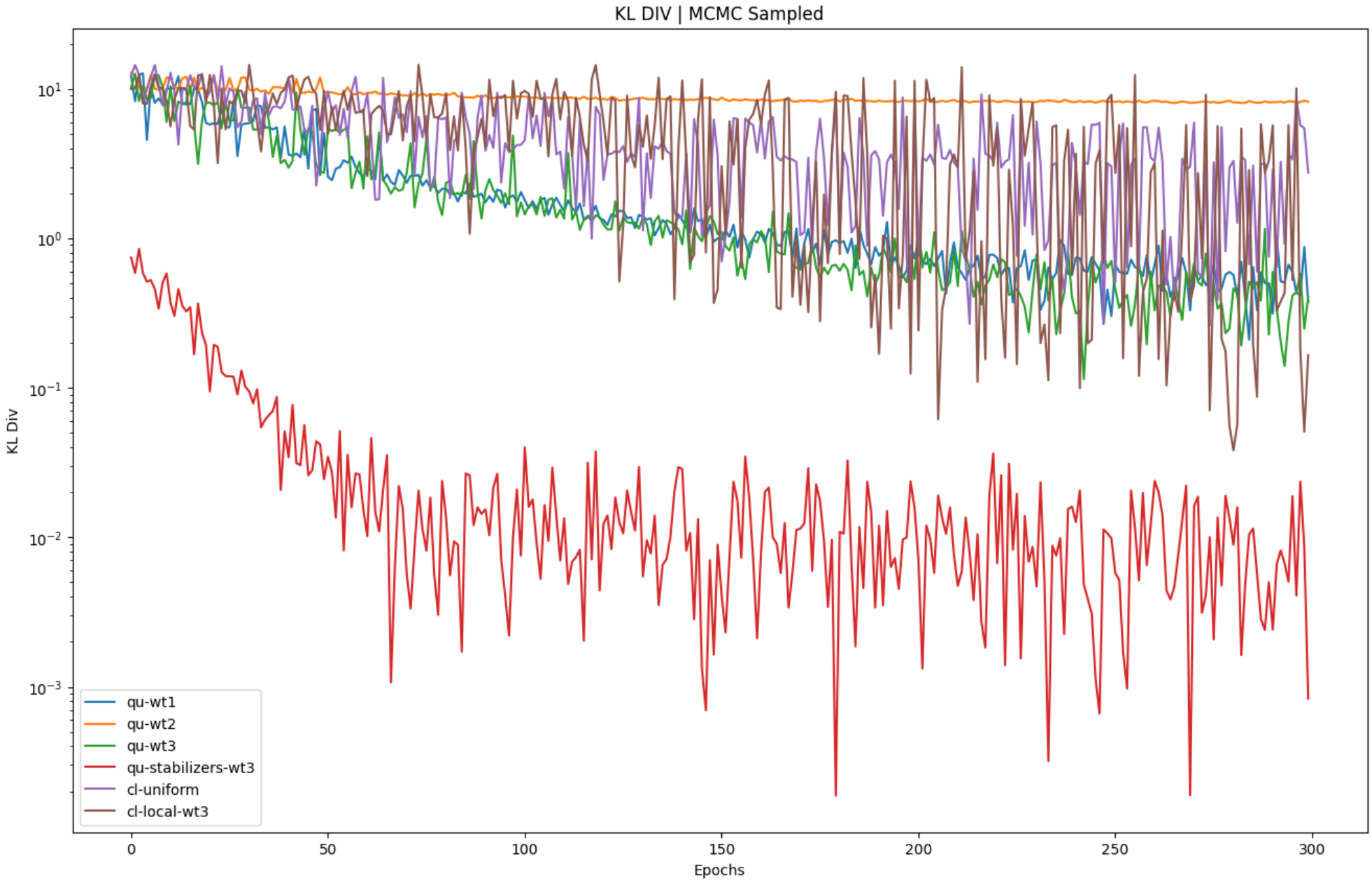
Results

KL Div



For the *'Exact-Sampled'* case the results are rather surprising. We see that the `cl-uniform` performing better than others mixers, and perhaps more interestingly we find that `qu-stabilizers-wt3` failing disastrously.





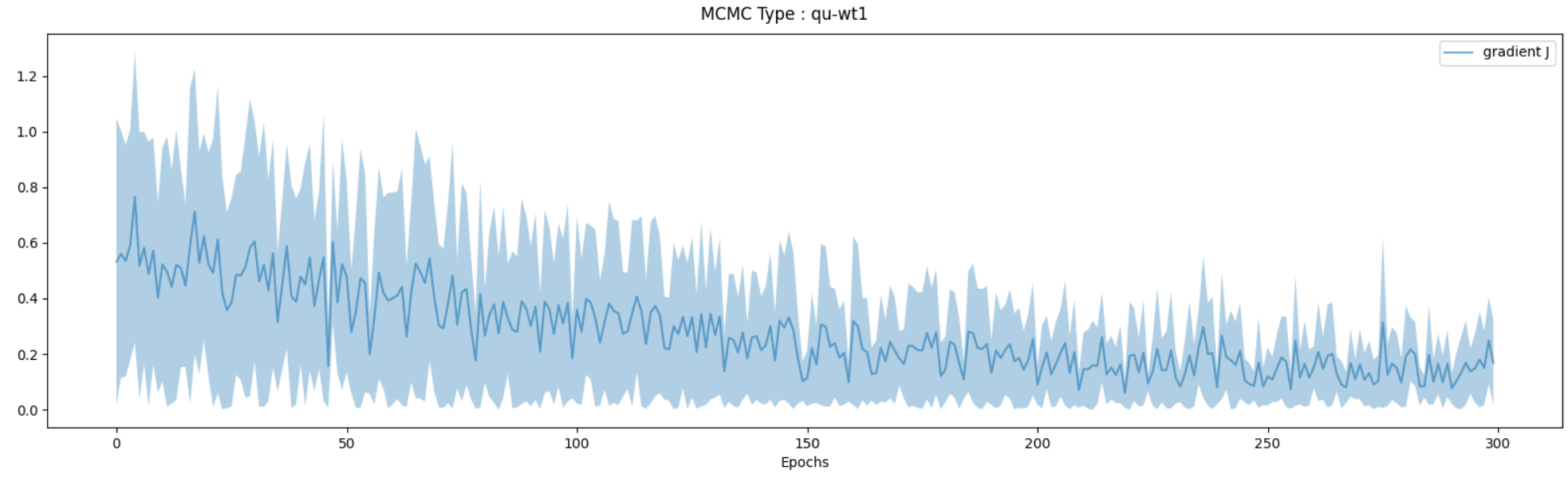
On the other hand *'MCMC Sampled'* computation of the KL-Div shows `qu-stabilizers-wt3` to have the significantly better performance than the rest. Among the rest, `qu-wt1` and `qu-wt3`
shows steady decrement at a comparable rate, while `cl-uniform` and `cl-local-wt1` shows unsteady and fluctuative behaviour throughout the training process. As expected `qu-wt2` has the worst performance.

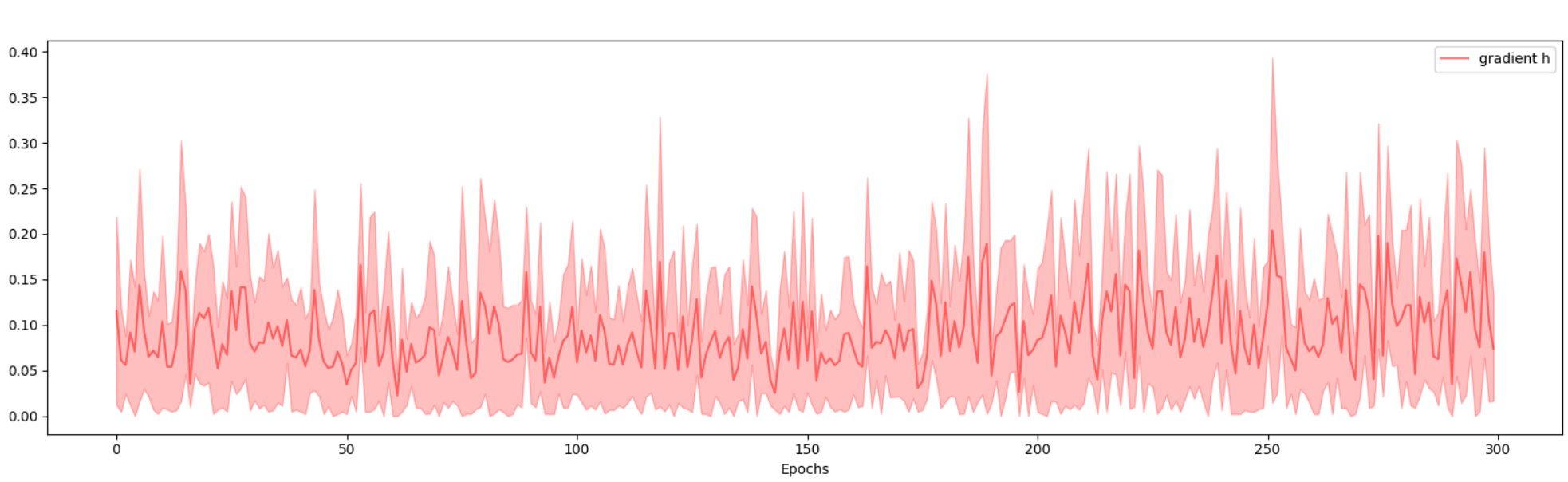
The contrast between the two cases is certainly concerning. And as I later realised, it has to do with the fact the fact that MCMC's with highly customised mixers (as in `qu-stabilizers-wt3`) fails miserably in the sampling process when the model to be trained is far from the target model. And since in training we start with a arbitrarily picked model, it drives the whole learning process in unintended directions.
To furhter elaborate the consequences of such mixers we will need to

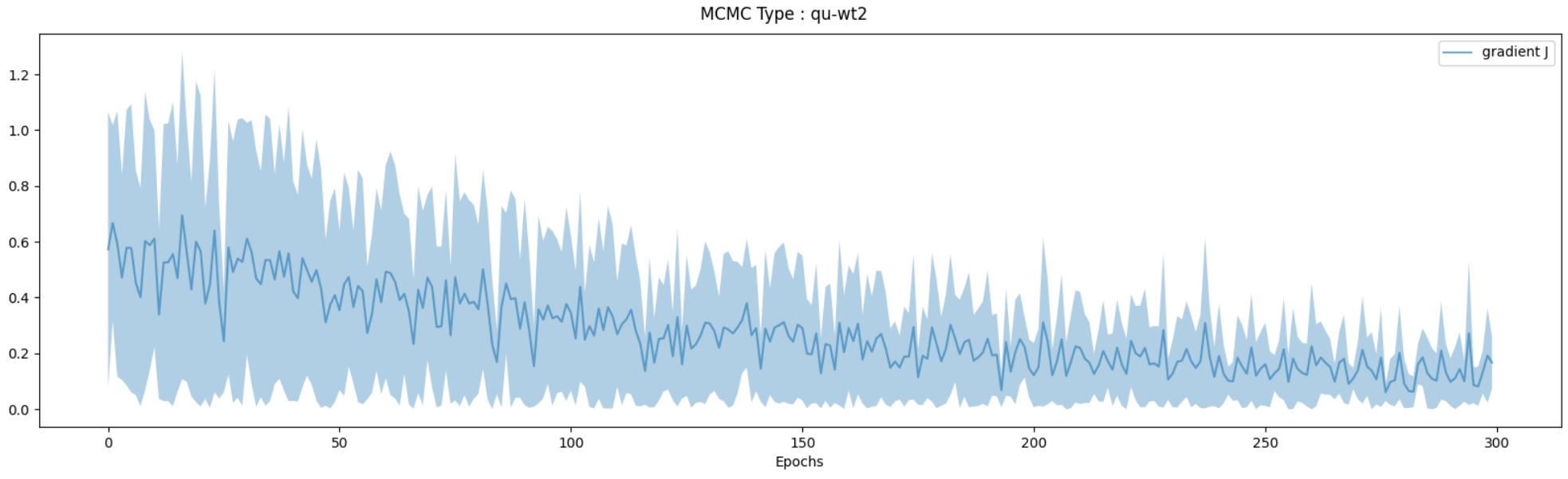
Gradient Variation

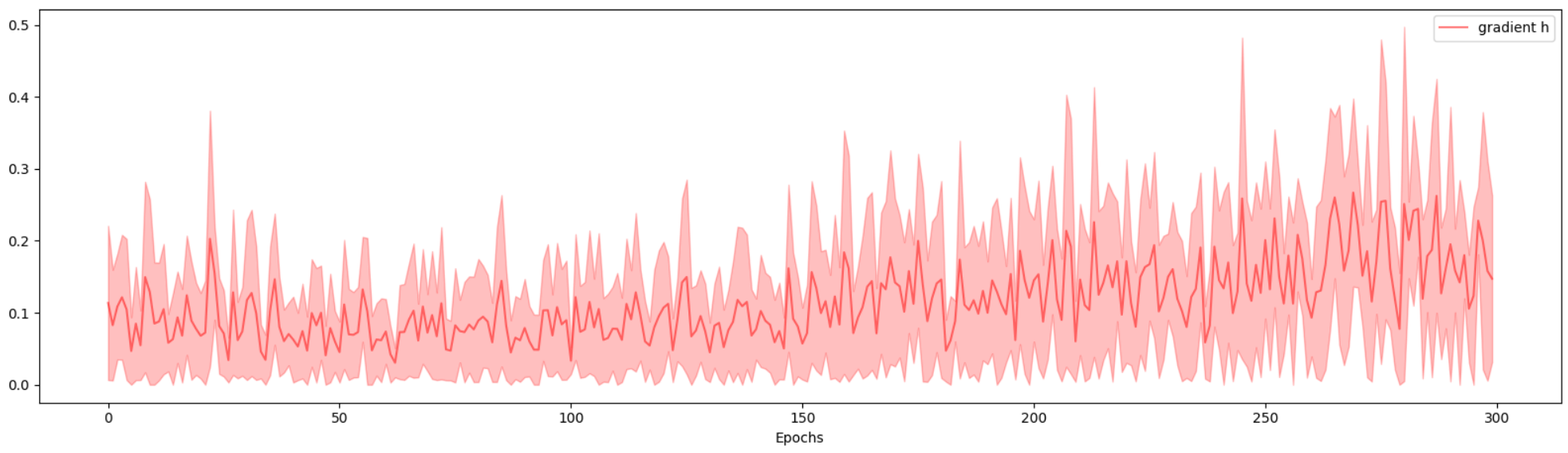
The plot tracks the variation of the max and min absolute values of the gradient that was computed in the each epoch.

Most of the cases shows similar variation in the parameters over the training rounds,

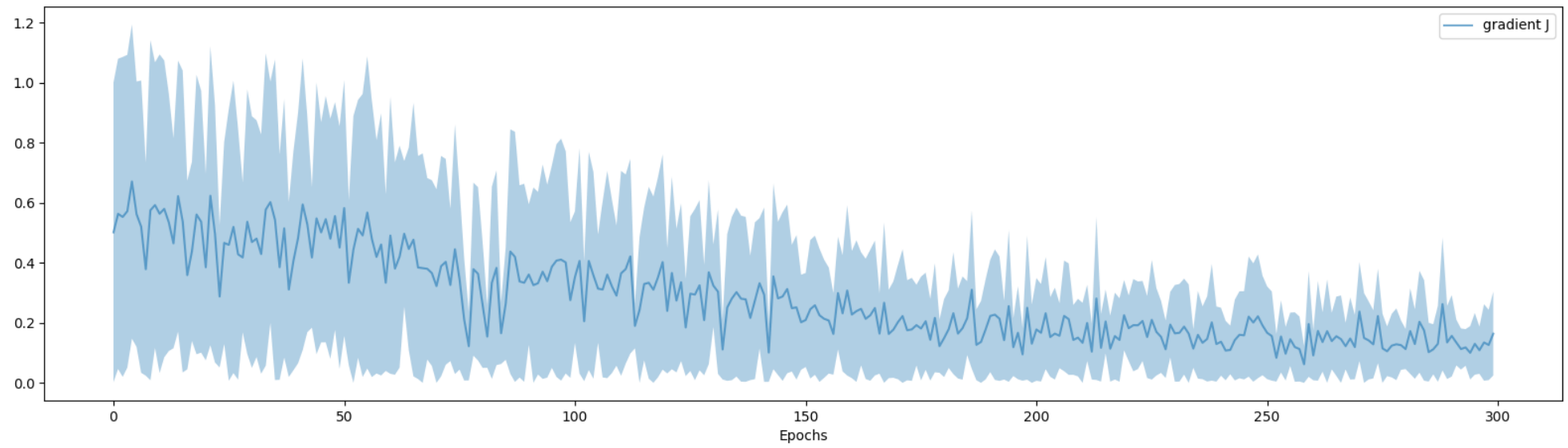


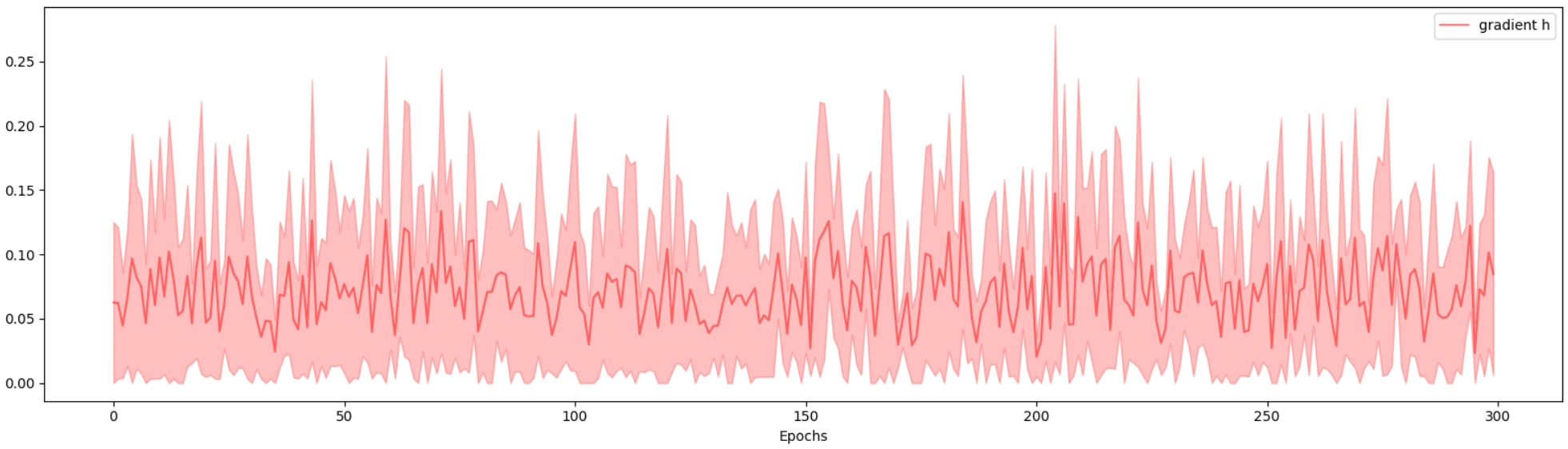




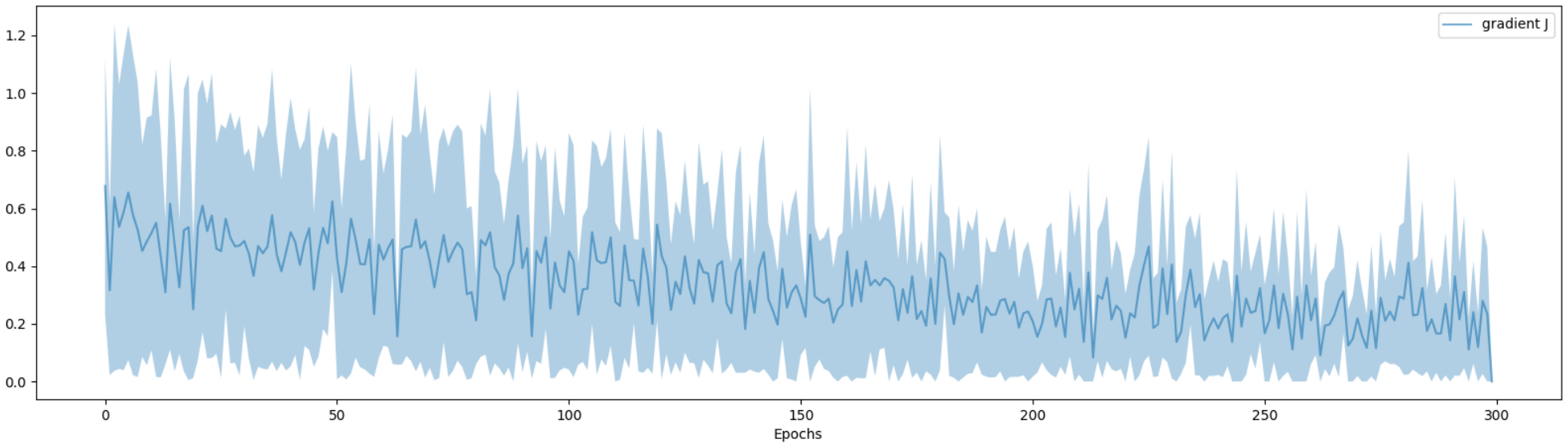


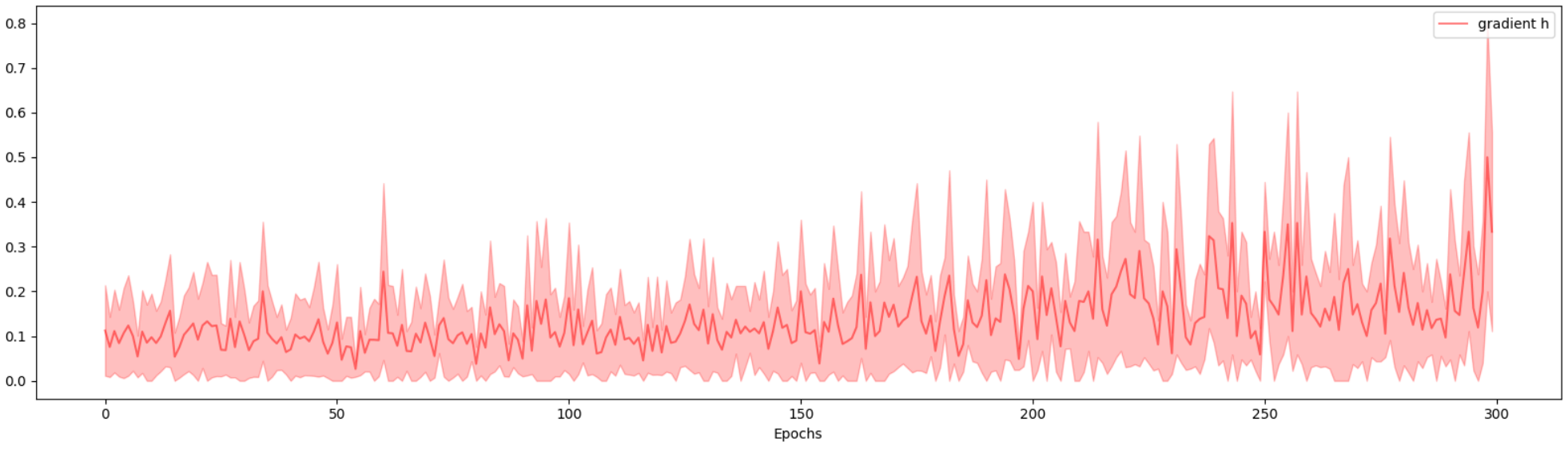
MCMC Type : qu-wt3

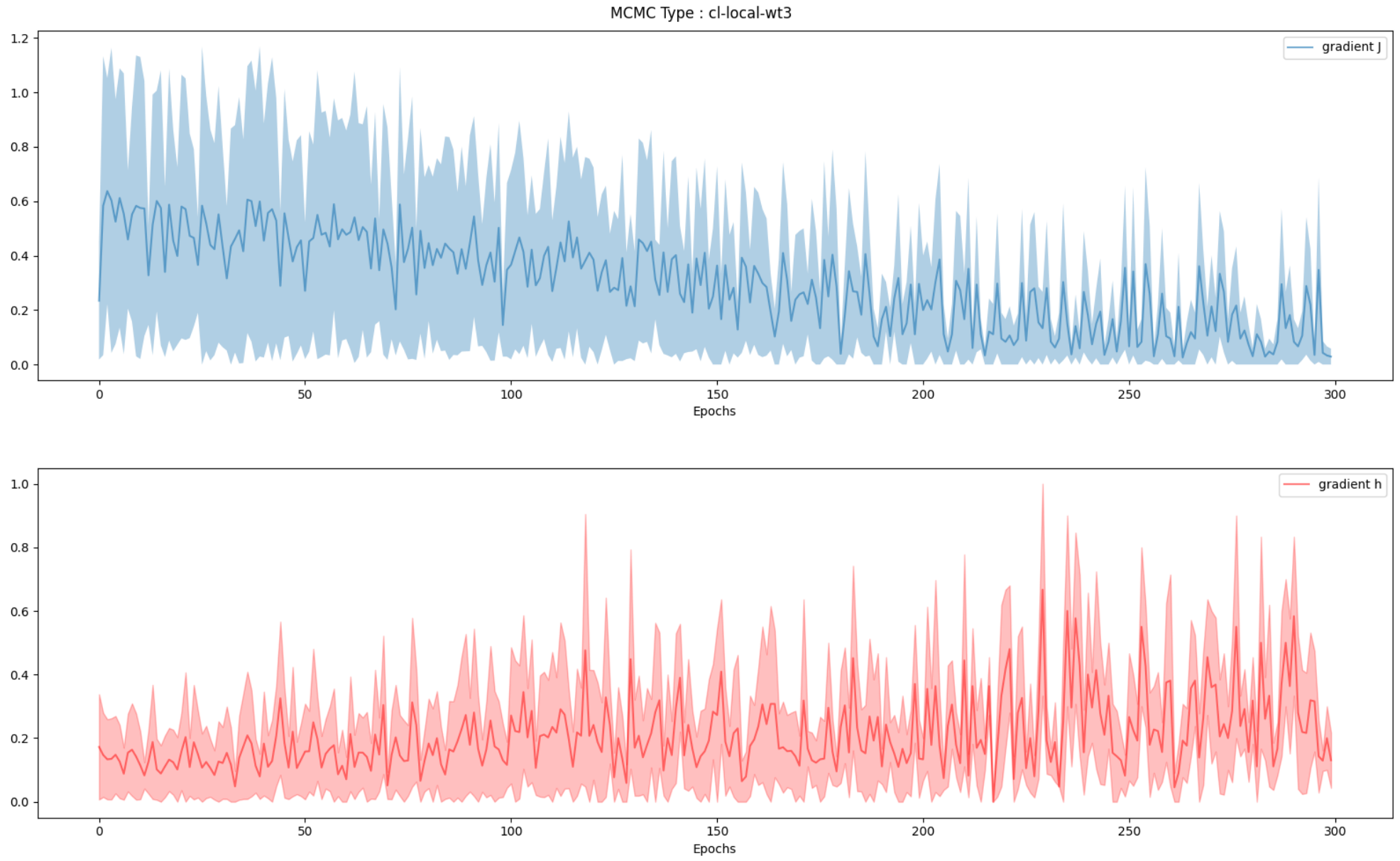




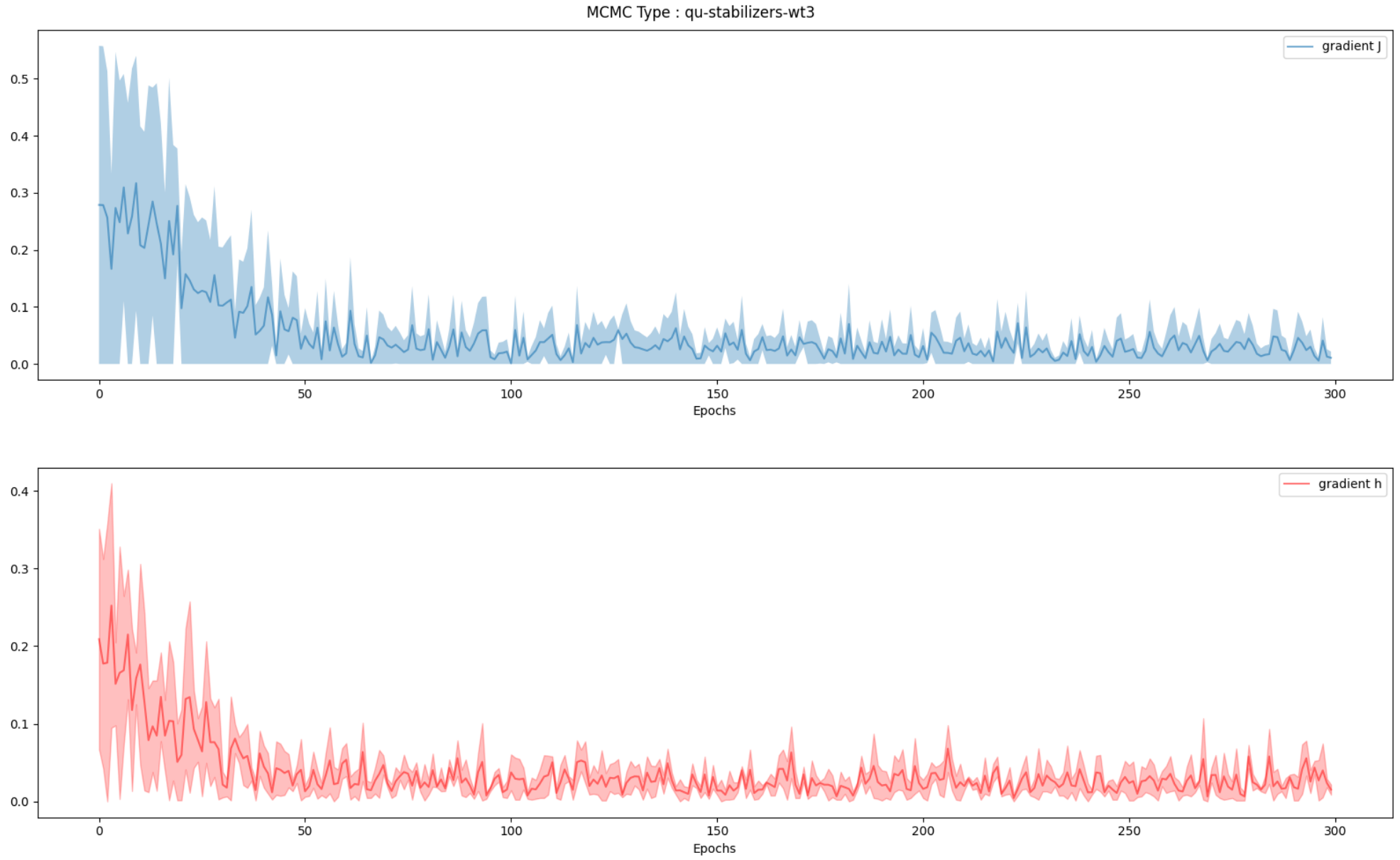
MCMC Type : cl-uniform







Except for the case of `qu-stabilizers-wt3` which shows significant distinction compared to rest of the cases,



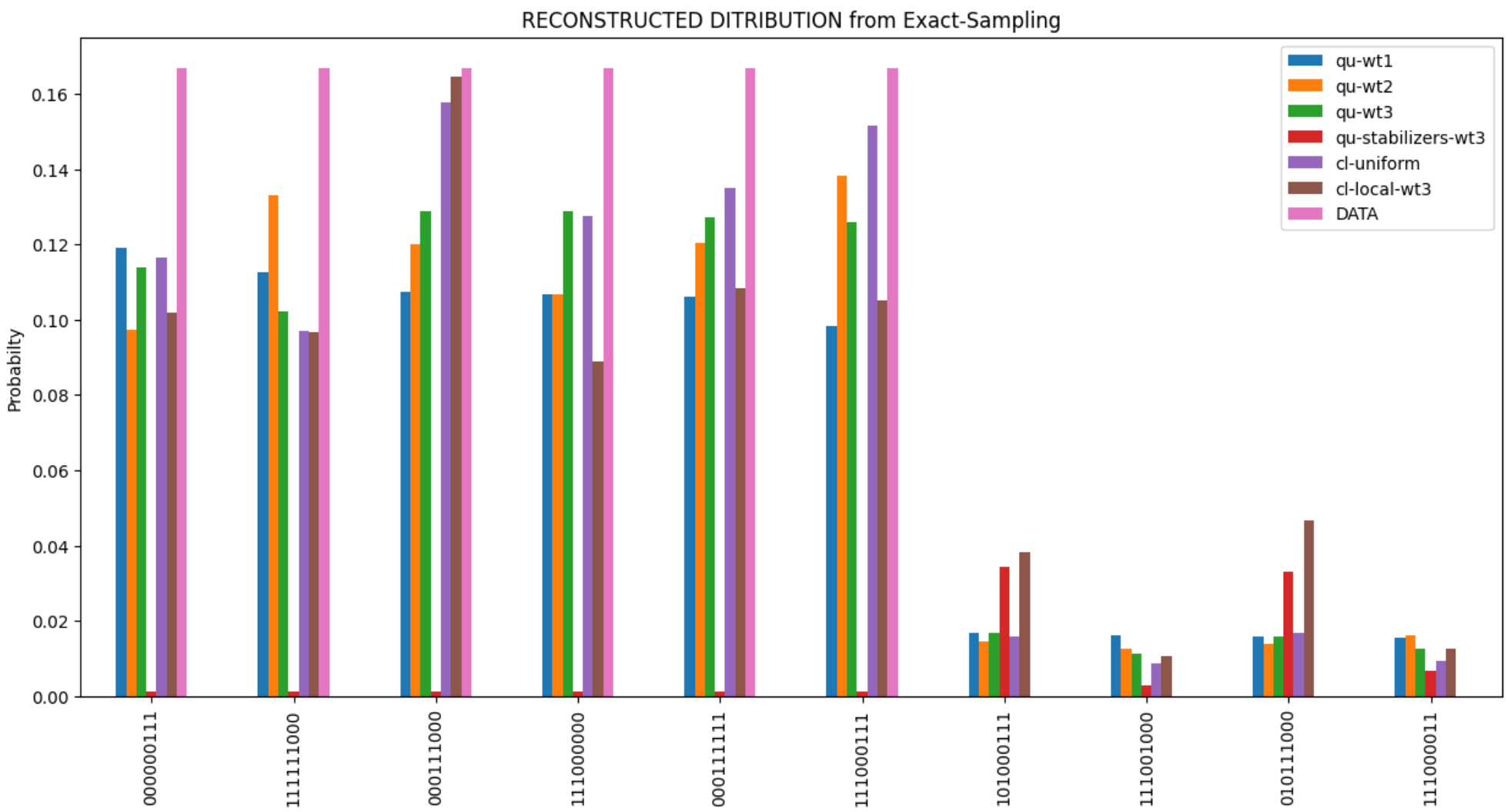
the plot indicates that gradient values saturate to the range $[0, 0.1]$ in about 100 epochs, is steep contrast to the other cases where the gradient fluctuates around $[0.1, 0.3]$ even after 300 epochs.

The gradient reflects the difference between expectation values of observables computed over the model and data, thus it is tempting to infer that MCMC sampling based on `qu-stabilizers-wt3` is more effective in the training process as the expectations calculated based on the sampler matches that of the data distribution. Above plots are certainly in line with the variation of KL-Div calculated over MCMC-Sampling, but contradicts the KL-Div variation calculated over Exact-Sampling.

To further investigate the reason for this discrepancy I tried directly comparing distribution generated by the model via Exact-Sampling and MCMC-Sampling.

Reconstructed Distribution

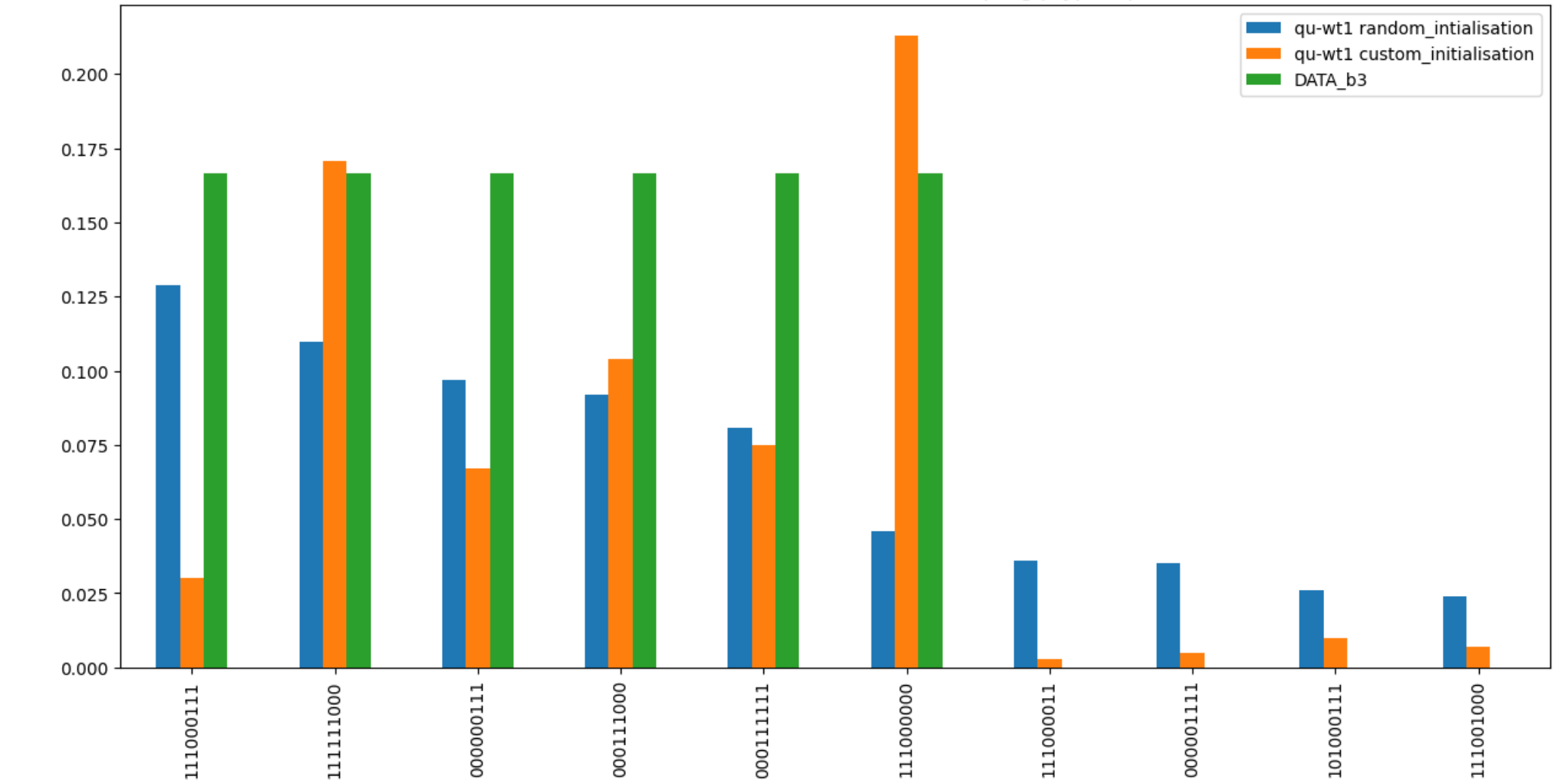
Exact-Sampling



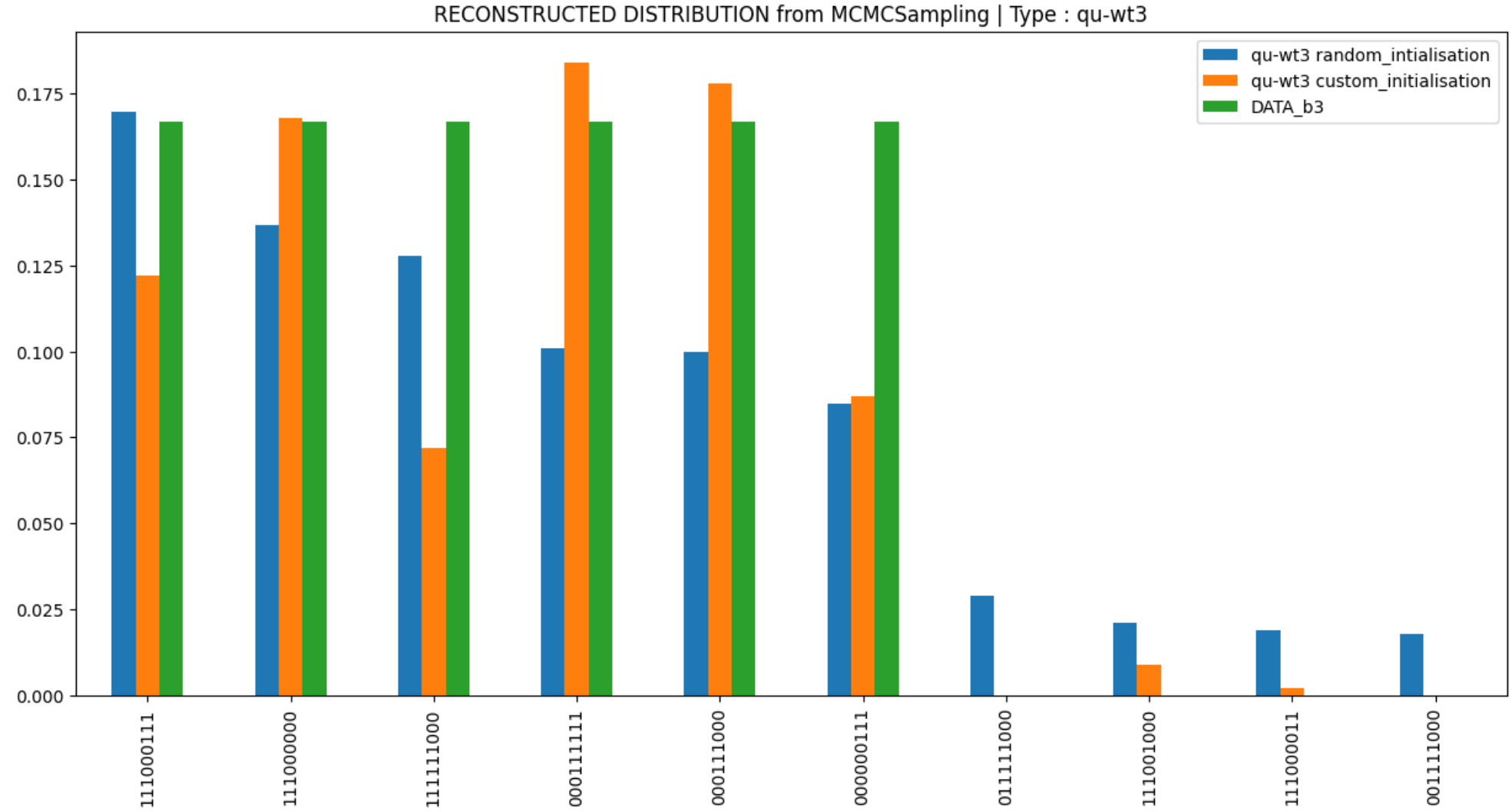
MCMC-Sampling

I realised that the initial state introduces significant bias in the trajectory of the markov chain for the cases where we customised the mixer to the data. This is certainly not surprising by itself, the mixers were designed with the aim to promote certain transitions over the other, but as we will see here constraining the transitions can in some cases adversely effect the training process.

I ran both the cases for a better comparison, one with an arbitrary initial state and another with an element of the dataset

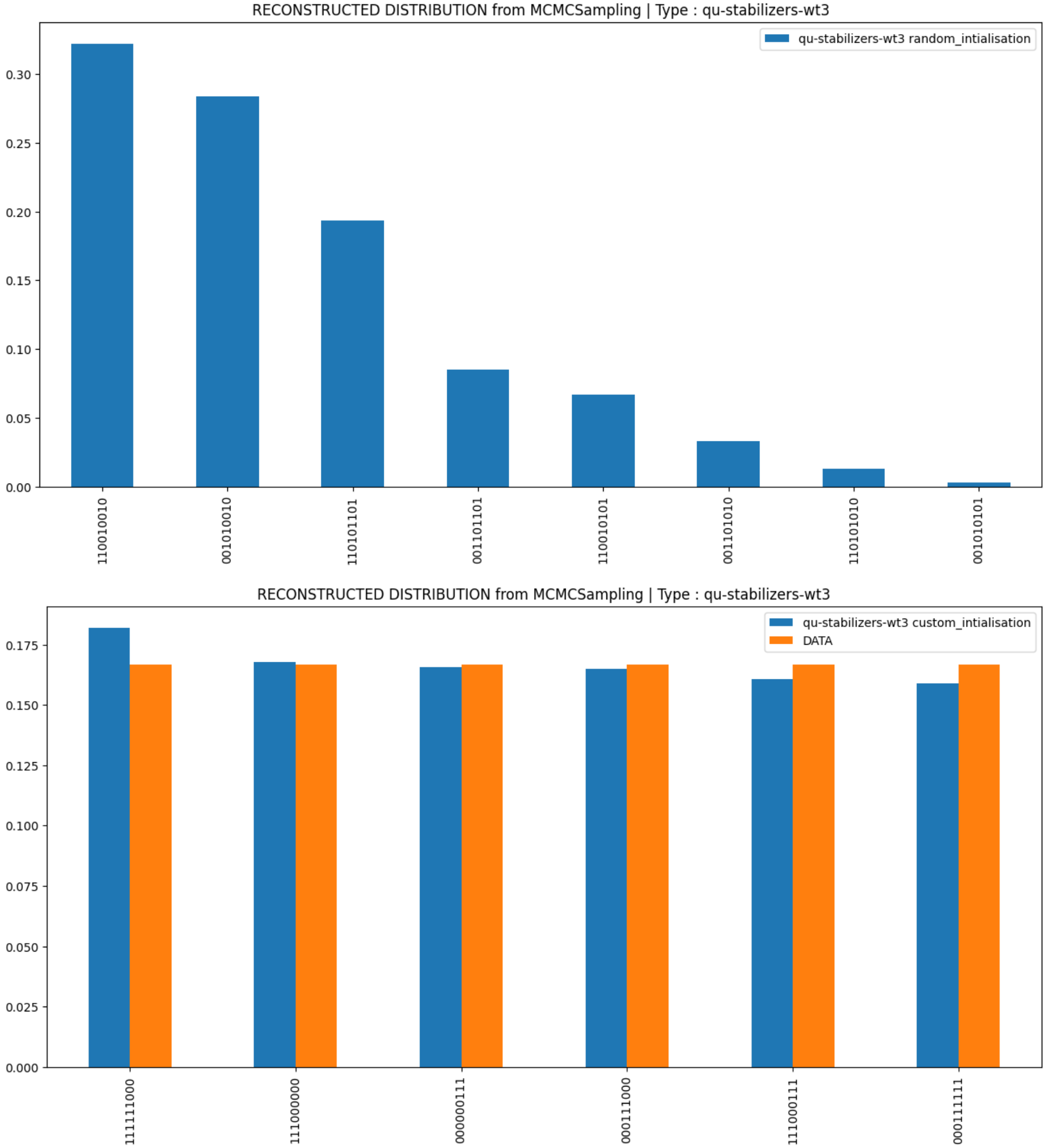

RECONSTRUCTED DISTRIBUTION from MCMCSampling | Type : qu-wt1
qu-wt1 random_intialisation
qu-wt1 custom_initialisation
DATA_b3
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
111000111
111111000
000000111
000111000
000111111
111000000
111000011
000001111
101000111
111001000


RECONSTRUCTED DISTRIBUTION from MCMCSampling | Type : qu-wt2
qu-wt2 random_intialisation
qu-wt2 custom_initialisation
DATA_b3
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
000111000
111000000
000000111
111110000
000101111
000110111
111000011
110111000
000111110
010111111



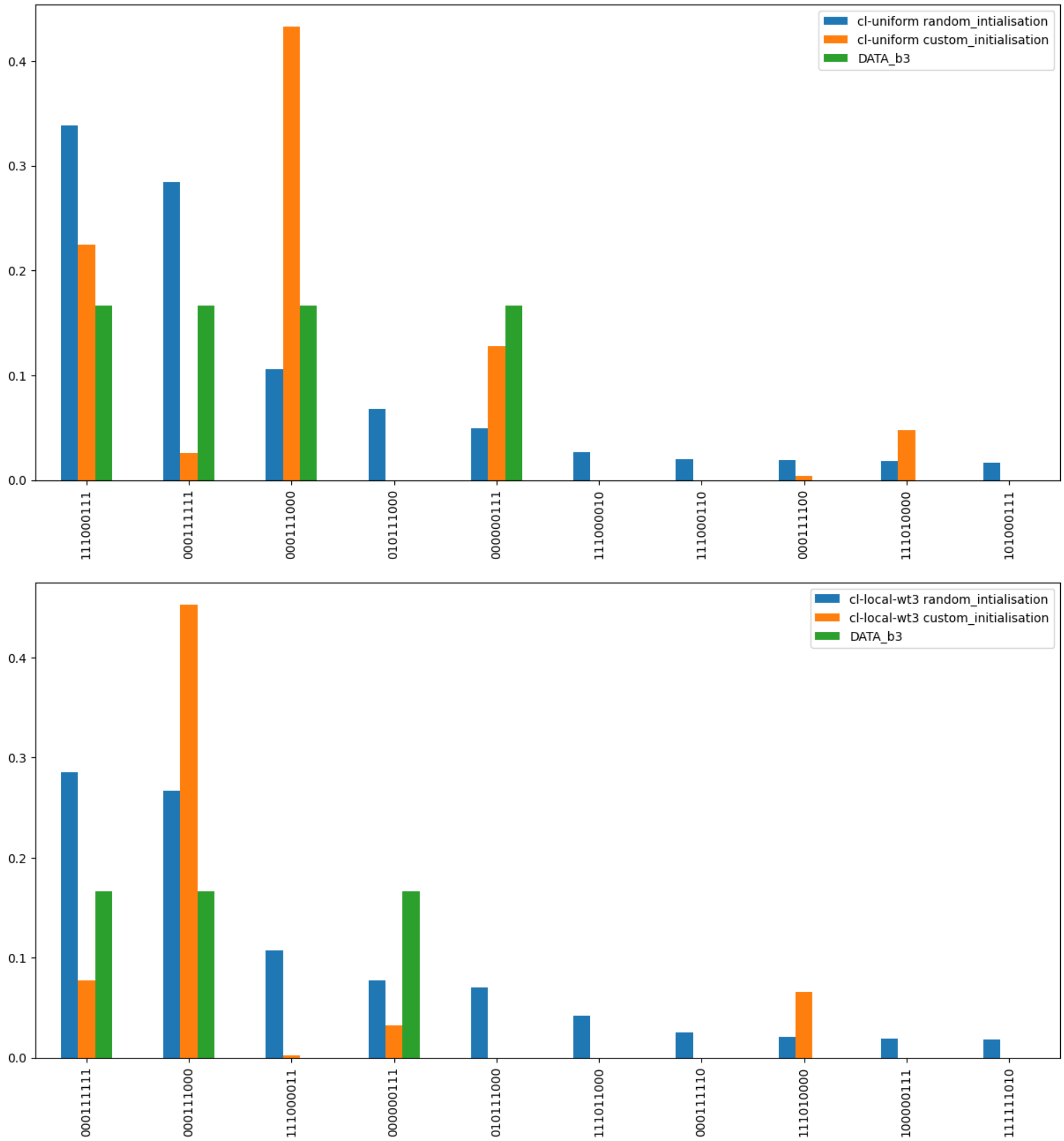
Expectedly `qu-wt2` performs badly, other two has moderate performance, however the effect of initialisation is quiet discernible in the case of `qu-wt3` .
The observation follows from the fact that `qu-wt3` imposes stronger constraints on the transitions and a markov chain with a wrong initialisation spends longer time before hitting a ground state, and the same mixers that facilitate ground-state to ground-state transitions hinders the algorithm to propose transitions to a ground state from a non-ground state. When we are sampling a fixed hamiltonian this might not have a significant effect but when the hamiltonian to be sampled is not fixed, as in the case with boltzmann-training, using constraint preserving mixers can have **'devastating'** effects.

Consider the case of `qu-stabilizers-wt3` ,



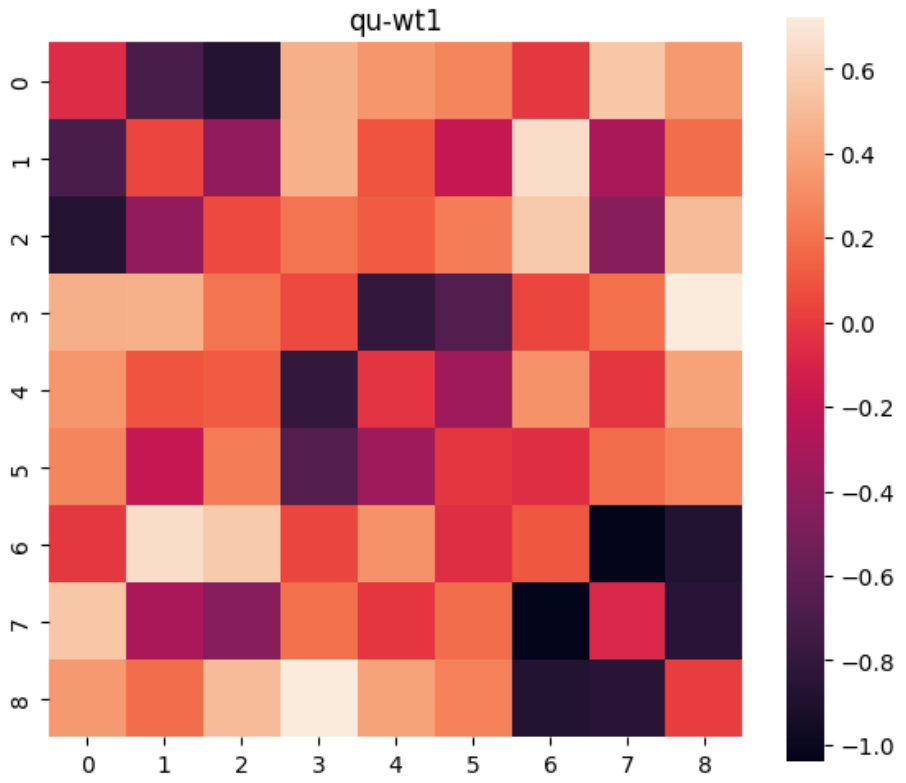
For a random-initialisation, the markov chain fails to sample either of the ground states whereas when initialised with an element from the dataset it returns a distribution almost indistinguishable to the data distribution.
Now this discrepancy should have came up our if we tried running sampling with random initialisation, but we can see its effect on the training process more directly by looking at the model parameters.

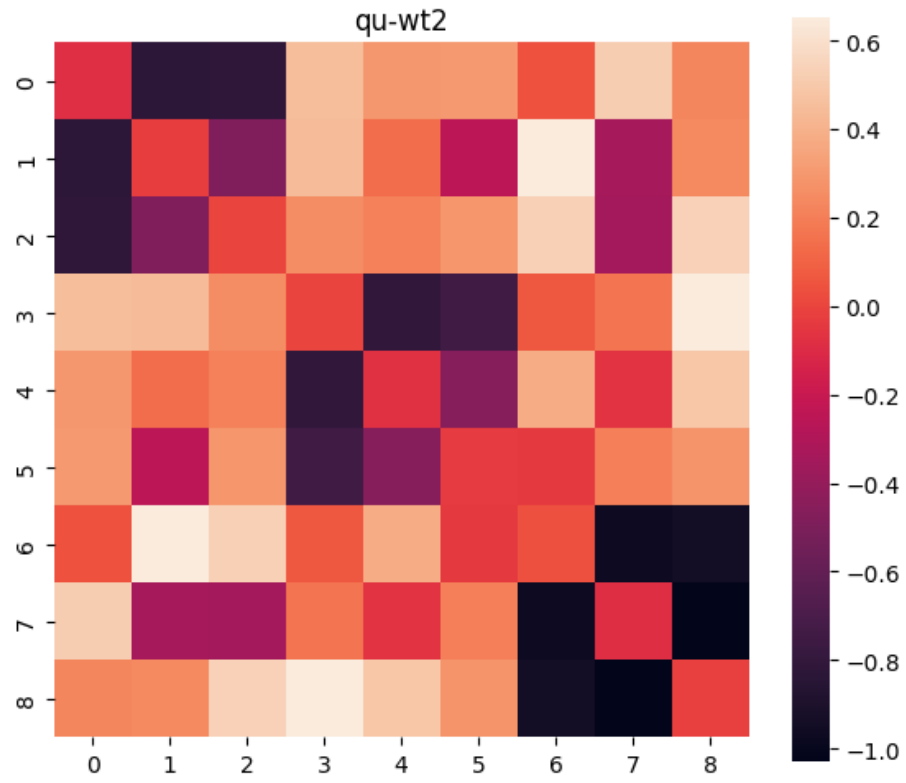
The classical cases under-perform comparatively.

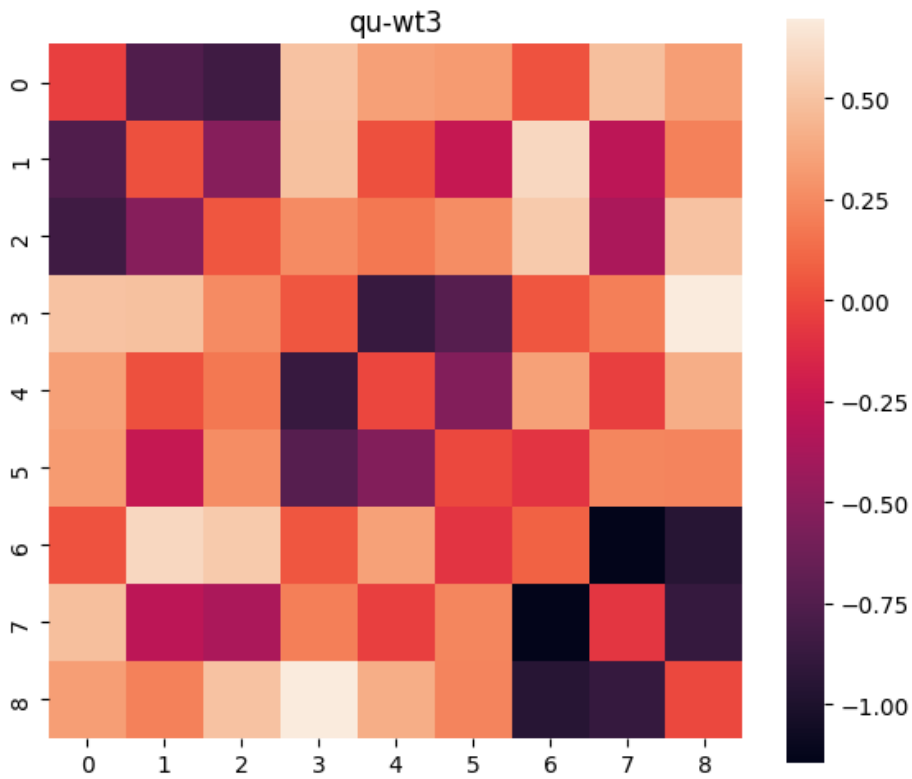


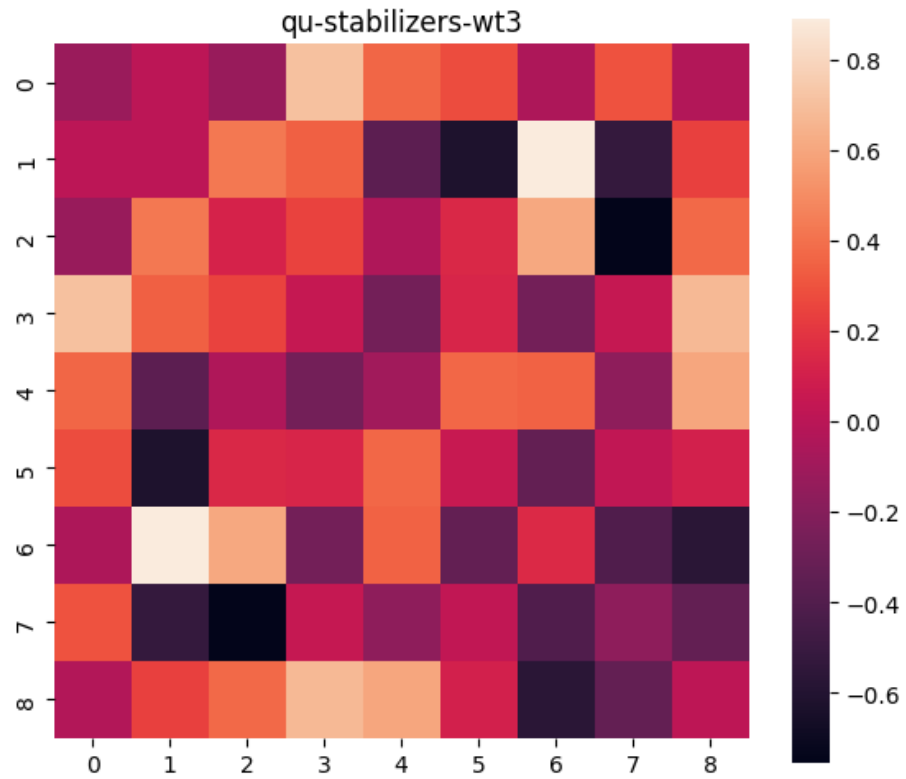
Parameter Visualisation

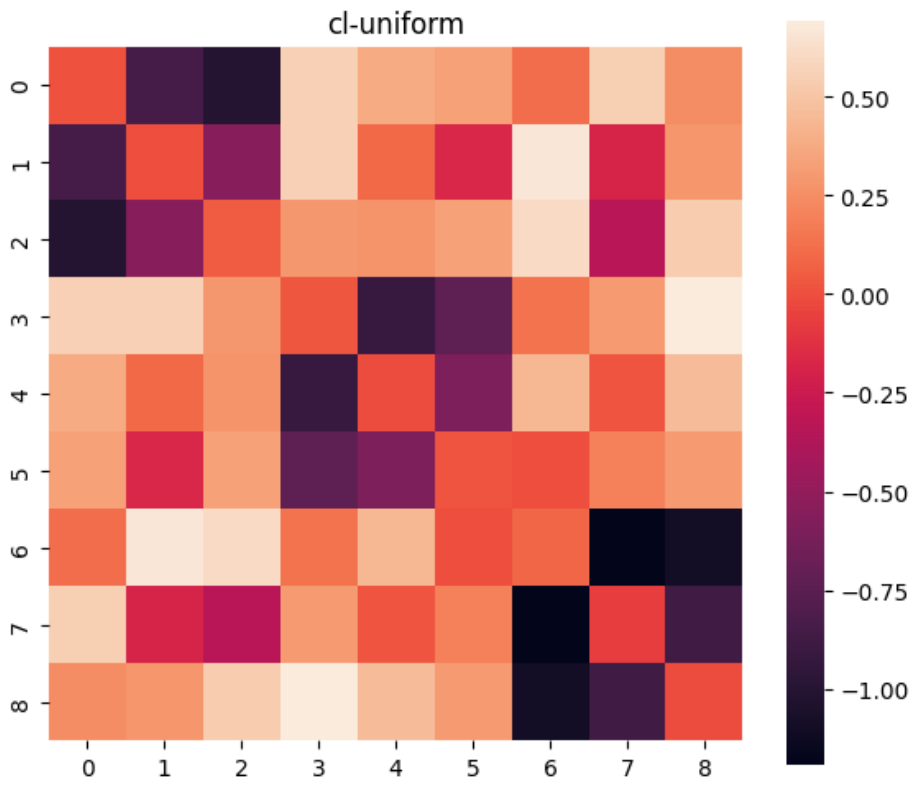
Here I have put out a visualisation for the training parameters of the trained model.

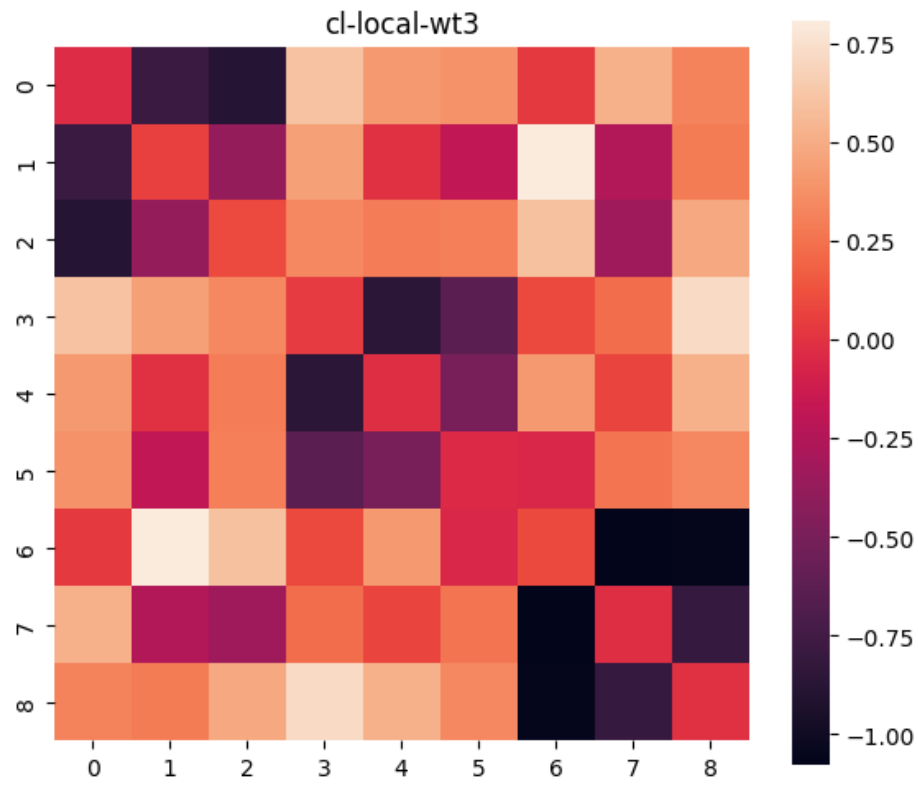












Comparing to the actual hamiltonian we can see the most models have manged to learn the "Stabalizer Terms" i.e the interaction like $\{-Z_0Z_1, -Z_1Z_2, -Z_0Z_2, \dots\}$ which are responsible for distinguishing the ground states, (you can check those 3×3 blocks of negative coefficients along the diagonal)

However notice that the model learnt by `qu-stabilizers-wt3` lies nowhere close to the rest of the models or to the target model for that matter. Given this erroneous model obtained through training it is surprising to know that it showed steady decrement in KL Divergence (at least in the MCMC-Sampled case) as well as a faster decrement of the gradient.

My hypothesis in this regard is that, the well structured mixer $T_{D^{b3}}$ assists the algorithm to exclusively sample the states in the dataset $\mathcal{D}^{b3}$, even when the inherent model is far from the target model.
Since we start the sampling from an element in the dataset the trajectory is remains confined within the dataset as well, and thus the expectation values computed tend to that of the data distribution after only few iterations of the training process. This explains the fast saturation of the gradient towards zero, and the fact that the parameters are barely updated to their target values.