R1, 7/10 *Although the maximum speedups over [18] in matrix inversion is quite sizable (27x in Fig 1), the gain over the much simpler, standard baseline seems moderate (2.7x to 4.3x). This potentially weakens the significance of the contributions, unless the proposed approach has advantages in other dimensions compared to the baseline.*

Indeed, the approach has advantages in other dimensions. FastH attains 5x to 27x speed-up for orthogonal matrices as used by [4,6,7,8,10,14,15,16,18], i.e, we can expect 5x to 27x if we use FastH in the referenced approaches (Figure 3). In particular, we found FastH to be so much faster than previous methods, that it can speed up even matrix inversion, something that was not possible with previous methods. We updated the introduction to reflect this distinction.

R4, 5/10 *The cost of searching proper k is $O(d^3)$ which is 'negligible' which makes me confused about which is not negligible in designed algorithm.*

Thanks for raising this concern. The default $k = batch\_size$ used in our experiments works well and one does not need to search for k. However, one might be able to improve slightly by trying different k. This only needs to be done **once** for a given hardware setup. This differs to our algorithm, FastH, which is used **every step during training**, i.e., FastH is used 10^5 times while the search is done 1 time. We rephrased a few sentences in section 3.3 to clarify this confusion.

The result in section 4.2, the sequential method spend too much time on inner products isn't mentioned in section 3.

We opted to use Section 3 to introduce our FastH algorithm, while clarifying the mentioned issue in the introduction (L22-28) and the background section (L66-70). We believe the current organization allows for a sharper separation between previous work and our proposed algorithm FastH.

There is a gap between theoretical time-complexity and empirical time-complexity which makes the analysis of time-complexity in section 3 can't support the effectiveness of the designed algorithm.

We believe there is a misconception here. "FastH retains the **same desirable time complexity** as the sequential algorithm from [18] while reducing the number of sequential operations" (introduction L31-33). In other words, FastH is $27x$ faster than [18] due to less sequential work, **not** due to a difference in time complexity.

R2, 6/10 *Limited applicability: seems the technique applies only to layers whose #input neurons = #output neurons.*

Thanks for raising this concern. Our technique does apply when the number of input neurons n is different to number of output neurons m, that is, for a regular linear layer with weight matrix $W \in \mathbb{R}^{n\times m}$. The weight matrix has a singular value decomposition $W = U\Sigma V^T$ for orthogonal U, V where $U \in \mathbb{R}^{n\times n}$, $V \in \mathbb{R}^{m\times m}$ and diagonal $\Sigma \in \mathbb{R}^{n\times m}$. FastH works for both U and V. This can furthermore be extended to attain semi-orthogonal W. [1] We added a paragraph in the subsection concerning extensions with the hopes that it clarifies this confusion.

The empirical methodology is a bit problematic since it does not explore deep learning at all, but rather on the time to do a single step and the cost of various matrix operations.

Thanks for raising this concern. The use of Householder matrices in deep learning has received much attention in previous work, e.g., [6,10,14,16,18]. FastH computes **exactly the same** as the algorithm used by [6,10,14,16,18]; repeating their experiments with FastH would thus attain the same results, albeit faster. Since we believe [6,10,14,16,18] adequately demonstrate the usefulness of Householder matrices for deep learning, we found the additional value of more such experiments were not that high. Furthermore, there are additional benefits to studying the performance of single operators as opposed to end-to-end deep learning experiments. Time complexity is a more transparent measure to investigate than the validation loss of deep learning models. Such measure shows the benefits of our approach **irrespective** of the architecture, optimizer, loss function and the many hyperparameters of complex networks.

Novelty: There is very little in the way of a fundamentally new idea. ... . The authors simply adjust the tool to the job.

As evidence against the claimed lack of novelty, we present three articles that would benefit from a $O(dm^2)$ parallel algorithm but did not "simply adjust the tool to the job." [18] realized the issue with sequential computation and suggested a parallel $O(d^3)$ algorithm. [10] uses the related CWY decomposition for gradient computations, without realizing WY can increase GPU utilization. They instead speed up by using fewer Householder matrices. An article contemporaneous[2] to our submission (`https://arxiv.org/abs/2004.08675`) addresses parallel Householder products with the related CWY decomposition, but attains a $O(d^3)$ bound (see their table 2, serial complexity is L^3).

R3, 7/10. *I would like to ask the authors to provide more details regarding the experimental setup to help reproducibility.*

We will soon open-source "neuralsvd.py" from the supplementary material, which we updated to run our main experiment and draw Figure 3. We also updated "README.txt" to contain more details regarding the experimental setup.

---

[1] Semi-orthogonal means $W^TW = I \neq WW^T$. This is true if $n > m$ and $\Sigma_{ii} = 1$.

[2] Contemporaneous, as per NeurIPS guidelines, refers to a work published less than two months before the submission deadline of which the authors are righteously not aware.