# Natural continual learning: success is a journey, not (just) a destination

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Biological agents are known to learn many different tasks over the course of their lives, and to be able to revisit previous tasks and behaviors with little to no loss in performance. In contrast, artificial agents are prone to 'catastrophic forgetting' whereby performance on previous tasks deteriorates rapidly as new ones are acquired. This shortcoming has recently been addressed using methods that encourage parameters to stay close to those used for previous tasks. This can be done by (i) using specific parameter regularizers that map out suitable destinations in parameter space, or (ii) guiding the optimization journey by projecting gradients into subspaces that do not interfere with previous tasks. However, parameter regularization has been shown to be relatively ineffective in recurrent neural networks (RNNs), a setting relevant to the study of neural dynamics supporting biological continual learning. Similarly, projection based methods can reach capacity and fail to learn any further as the number of tasks increases. To address these limitations, we propose Natural Continual Learning (NCL), a new method that unifies weight regularization and projected gradient descent. NCL uses Bayesian weight regularization to encourage good performance on all tasks at convergence and combines this with gradient projections designed to prevent catastrophic forgetting during optimization. NCL formalizes gradient projection as a trust region algorithm based on the Fisher information metric, and achieves scalability via a novel Kronecker-factored approximation strategy. Our method outperforms both standard weight regularization techniques and projection based approaches when applied to continual learning problems in RNNs. The trained networks evolve task-specific dynamics that are strongly preserved as new tasks are learned, similar to experimental findings in biological circuits.

## 1 Introduction

Catastrophic forgetting is a common feature of machine learning algorithms where training on a new task often leads to poor performance on previously learned tasks. This is in contrast to biological agents which are capable of learning many different behaviors over the course of their lives with little to no interference across tasks. The study of continual learning in biological networks may therefore help inspire novel approaches in machine learning, while the development and study of continual learning algorithms in artificial agents can help us better understand how this challenge is overcome in the biological domain. This is particularly true in recurrent neural networks (RNNs) which are important due to their practical and biological relevance. However, continual learning in RNNs has recently proven challenging for many existing algorithms [8, 9]. For these reasons, we focus our experiments on RNNs in the present work although our algorithm and theoretical considerations are applicable to any continual learning setting that can be formalized as a probabilistic model, whether the problem is supervised or unsupervised and the architecture recurrent or feedforward.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Previous work has addressed the challenge of continual learning in artificial agents using weight regularization where parameters important for previous tasks are regularized to stay close to their previous values [12, 17, 25, 29]. This approach can be motivated by findings in the neuroscience literature of increased stability for a subset of synapses after learning [35, 36]. More recently, approaches based on projecting gradients into subspaces orthogonal to those that are important for previous tasks have been developed in both feedforward [39] and recurrent [8] neural networks. This is consistent with experimental findings that neural dynamics often occupy orthogonal subspaces across contexts in biological circuits [2, 15]. While these methods have been found to perform well in many continual learning settings, they also suffer from various shortcomings. In particular, while Bayesian weight regularization provides a natural way to weigh previous and current task information, this approach can fail in practice due to its approximate nature and often requires additional tuning of the importance of the prior beyond what would be expected in a rigorous Bayesian treatment [33]. In contrast, while projection based methods have been found empirically to mitigate catastrophic forgetting, it is unclear how the 'important subspaces' should be selected and how such methods behave when task demands begin to saturate the network capacity.

In this work, we develop a new method for continual learning, natural continual learning (NCL), by combining Bayesian continual learning using weight regularization with an optimization procedure that relies on a trust region constructed from an approximate posterior distribution over the parameters given previous tasks. This encourages parameter updates predominantly in the null-space of previously acquired tasks while maintaining convergence to maxima of the Bayesian approximate posterior. We show that NCL outperforms previous continual learning algorithms, and that our principled Bayesian treatment is particularly important when task number and complexity increases or network size decreases. We also show that the projection based methods introduced by Duncker et al. [8] and Zeng et al. [39] can be viewed as approximations to such trust region optimization using the posterior from previous tasks. Finally we use tools from the neuroscience literature to investigate how the learned networks overcome the challenge of continual learning. Here we find that the networks learn latent task representations that are stable over time after initial task learning, consistent with results from biological circuits.

## 2 Method

**Notations** We use $\boldsymbol{X}^\top$, $\boldsymbol{X}^{-1}$, $\mathrm{Tr}(\boldsymbol{X})$ and $\mathrm{vec}(\boldsymbol{X})$ to denote the transpose, inverse, trace, and column-wise vectorization of a matrix $\boldsymbol{X}$. We use $\boldsymbol{X} \otimes \boldsymbol{Y}$ to represent the Kronecker product between matrices $\boldsymbol{X} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{Y} \in \mathbb{R}^{m\times m}$ such that $(\boldsymbol{X} \otimes \boldsymbol{Y})_{mi+k,mj+l} = \boldsymbol{X}_{ij}\boldsymbol{Y}_{kl}$. We use bold lower-case letters $\boldsymbol{x}$ to denote column vectors. We use 'OWM' to refer to orthogonal weight modification [39] and 'DOWM' for 'doubly orthogonal weight modification' to refer to the method proposed by Duncker et al. [8]. $\mathcal{D}_k$ refers to a 'dataset' corresponding to task $k$ which in this work generally consists of a set of input-output pairs $\{\boldsymbol{x}_k^{(i)}, \boldsymbol{y}_k^{(i)}\}$ such that $\ell_k(\boldsymbol{\theta}) := \log p(\mathcal{D}_k|\boldsymbol{\theta}) = \sum_i \log p_{\boldsymbol{\theta}}(\boldsymbol{y}_k^{(i)}|\boldsymbol{x}_k^{(i)})$ is the task-related performance on task $k$ for a model with parameters $\boldsymbol{\theta}$. Finally, we use $\hat{\mathcal{D}}_k$ to refer to a dataset generated by inputs from the $k^{th}$ task where $\{\hat{\boldsymbol{y}}_k^{(i)} \sim p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}_k^{(i)})\}$ are drawn from the model distribution.

### 2.1 Bayesian continual learning

**Problem statement** In continual learning, we train a model on a set of $K$ datasets $\{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$ that arrive sequentially. The aim is to learn a probabilistic model $p(\mathcal{D}|\boldsymbol{\theta})$ that performs well on all tasks. The challenge in the continual learning setting stems from the sequential nature of learning, and in particular from the common assumption that the learner does not have access to "past" datasets (i.e., $\mathcal{D}_j$ for $j < k$) when learning task $k$. While we enforce this stringent condition in this paper, our approach may be easily combined with memory-based techniques such as coresets and generative replay which allow for storage of a subset of past data or an increase in model parameters with each task [9, 25, 27, 34].

**Bayesian approach** The continual learning problem is naturally formalized in a Bayesian framework whereby the posterior after $k-1$ tasks is used as a prior for task $k$. More specifically, we choose a prior $p(\boldsymbol{\theta})$ on the model parameters and compute the posterior after observing $k$ datasets

according to Bayes' rule:

$$p(\boldsymbol{\theta}|\mathcal{D}_{1:k}) \propto p(\boldsymbol{\theta}) \prod_{k'=1}^{k} p(\mathcal{D}_{k'}|\boldsymbol{\theta})$$
$$\propto p(\boldsymbol{\theta}|\mathcal{D}_{1:k-1}) p(\mathcal{D}_k|\boldsymbol{\theta}), \tag{1}$$

where $\mathcal{D}_{1:k}$ is a concatenation of the first $k$ datasets $(\mathcal{D}_1, \ldots, \mathcal{D}_k)$. In theory, it is thus possible to compute the exact posterior $p(\boldsymbol{\theta}|\mathcal{D}_{1:k})$ after $k$ datasets, while only observing $\mathcal{D}_k$, by using the posterior $p(\boldsymbol{\theta}|\mathcal{D}_{1:k-1})$ after $k-1$ tasks as a prior. However, as is often the case in Bayesian inference, the difficulty here is that the posterior is typically intractable. To address this challenge, it is common to perform approximate online Bayesian inference. That is, the posterior $p(\boldsymbol{\theta}|\mathcal{D}_{1:k-1})$ is approximated by a parametric distribution with parameters $\boldsymbol{\phi}_{k-1}$. The approximate posterior $q(\boldsymbol{\theta}; \boldsymbol{\phi}_{k-1})$ is then used as a prior for task $k$.

**Online Laplace approximation** A common approach is to use the Laplace approximation whereby the posterior $p(\boldsymbol{\theta}|\mathcal{D}_{1:k-1})$ is approximated as a multivariate Gaussian $q$ using local gradient information [12, 17, 29]. This involves (i) finding a mode $\boldsymbol{\mu}_k$ of the posterior during task $k$, and (ii) performing a second-order Taylor expansion around $\boldsymbol{\mu}_k$ to construct an approximate Gaussian posterior $q(\boldsymbol{\theta}; \boldsymbol{\phi}_k) = \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1})$, where $\boldsymbol{\Lambda}_k$ is the precision matrix and $\boldsymbol{\phi}_k = (\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k)$. In this case, gradient-based optimization is used to find the posterior mode on task $k$ (c.f. Equation 1):

$$\boldsymbol{\mu}_k = \arg\max_{\boldsymbol{\theta}} \ \log p(\boldsymbol{\theta}|\mathcal{D}_k, \boldsymbol{\phi}_{k-1}) \tag{2}$$
$$= \arg\max_{\boldsymbol{\theta}} \ \log p(\mathcal{D}_k|\boldsymbol{\theta}) + \log q(\boldsymbol{\theta}|\boldsymbol{\phi}_{k-1}) \tag{3}$$
$$= \arg\max_{\boldsymbol{\theta}} \ \underbrace{\ell_k(\boldsymbol{\theta}) - \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\mu}_{k-1})^\top \boldsymbol{\Lambda}_{k-1}(\boldsymbol{\theta} - \boldsymbol{\mu}_{k-1})}_{:= \mathcal{L}_k(\boldsymbol{\theta})} \tag{4}$$

The precision matrix $\boldsymbol{\Lambda}_k$ is given by the Hessian of the negative log posterior at $\boldsymbol{\mu}_k$:

$$\boldsymbol{\Lambda}_k = -\nabla^2_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\mathcal{D}_k, \boldsymbol{\phi}_{k-1})\big|_{\boldsymbol{\theta}=\boldsymbol{\mu}_k} = H(\mathcal{D}_k, \boldsymbol{\mu}_k) + \boldsymbol{\Lambda}_{k-1}, \tag{5}$$

where $H(\mathcal{D}_k, \boldsymbol{\mu}_k) = -\nabla^2_{\boldsymbol{\theta}} \log p(\mathcal{D}_k|\boldsymbol{\theta})\big|_{\boldsymbol{\theta}=\boldsymbol{\mu}_k}$ is the Hessian of the negative log likelihood of $\mathcal{D}_k$.

Continual learning with the online Laplace approximation thus involves two steps for each new dataset $\mathcal{D}_k$. First, given $\mathcal{D}_k$ and the previous posterior $q(\boldsymbol{\theta}|\boldsymbol{\mu}_{k-1}, \boldsymbol{\Lambda}_{k-1}^{-1})$ (i.e. the new prior), $\boldsymbol{\mu}_k$ is found using gradient-based optimization (Equation 4). This step can be interpreted as optimizing the likelihood of $\mathcal{D}_k$ while penalizing changes in the parameters $\boldsymbol{\theta}$ according to their importance for previous tasks, as determined by the prior precision matrix $\boldsymbol{\Lambda}_{k-1}$. Second, the new posterior precision matrix $\boldsymbol{\Lambda}_k$ is computed according to Equation 5.

**Approximating the Hessian** In practice, computing $\boldsymbol{\Lambda}_k$ presents two major difficulties. First, because $q(\boldsymbol{\theta}; \boldsymbol{\phi}_k)$ is a Gaussian distribution, $\boldsymbol{\Lambda}_k$ has to be positive semi-definite (PSD) which is not guaranteed for the Hessian $H(\mathcal{D}_k, \boldsymbol{\mu}_k)$. Second, if the number of model parameters $n_\theta$ is large, it may be prohibitive to compute a full $(n_\theta \times n_\theta)$ matrix. To address the first issue, it is common to approximate the Hessian with the Fisher information matrix (FIM; 12, 29):

$$\boldsymbol{F}_k = \mathbb{E}_{p(\hat{\mathcal{D}}_k|\boldsymbol{\theta})} \left[ \nabla_{\boldsymbol{\theta}} \log p(\hat{\mathcal{D}}_k|\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(\hat{\mathcal{D}}_k|\boldsymbol{\theta})^\top \right]\Big|_{\boldsymbol{\theta}=\boldsymbol{\mu}_k} \approx H(\mathcal{D}_k, \boldsymbol{\mu}_k) \tag{6}$$

The FIM is PSD which ensures that $\boldsymbol{\Lambda}_k = \sum_{k'=1}^{k} \boldsymbol{F}_{k'}$ is also PSD. Computing $\boldsymbol{F}_k$ may still be impractical if there are many model parameters, and it is therefore common to further approximate the FIM using structured approximations with fewer parameters. In particular, a diagonal approximation to $\boldsymbol{F}_k$ recovers Elastic Weight Consolidation (EWC; 17) while a Kronecker-Factored approximation [24] recovers the method proposed by Ritter et al. [29]. We denote this method 'KFAC' and use it in Section 3 as a comparison for our own Kronecker-factored method.

### 2.2 Natural continual learning

While the online Laplace approximation has been applied successfully in several continual learning settings [17, 29], it has also been found to perform sub-optimally on a range of problems [8, 33]. Additionally, its Bayesian interpretation in theory prescribes a unique way of weighting the contributions

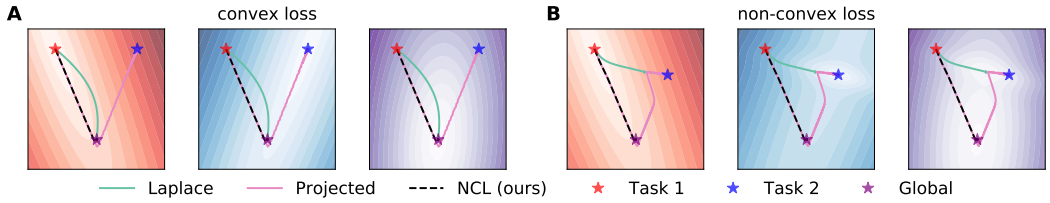


Figure 1: **Continual learning in a toy problem. (A)** Loss landscape on toy task 1 ($\ell_1$; left), task 2 ($\ell_2$; middle) and the combined loss $\ell_{1+2} = \ell_1 + \ell_2$ (right). Stars indicate the global optima for $\ell_1$ (red), $\ell_2$ (blue), and $\ell_{1+2}$ (purple). We assume that $\boldsymbol{\theta}$ has been optimized for $\ell_1$ and consider how learning proceeds on task 2 using either the Laplace posterior ('Laplace', green), projected gradient descent on $\ell_2$ with preconditioning according to task 1 ('Projected', pink), or NCL (black dashed). Laplace follows the steepest gradient of $\ell_{1+2}$ and transiently forgets task 1. NCL follows a flat direction of $\ell_1$ and converges to the global optimum of $\ell_{1+2}$ with good performance on task 1 throughout. Projected gradient descent follows a similar optimization path to NCL but eventually diverges towards the optimum of $\ell_2$. **(B)** As in (A), now with non-convex $\ell_2$ (center), leading to a second local optimum of $\ell_{1+2}$ (right) while $\ell_1$ is unchanged (left). In this case, Laplace can converge to a local optimum which has 'catastrophically' forgotten task 1. Projected gradient descent moves only slowly in 'steep' directions of $\ell_1$ but eventually converges to a minimum of $\ell_2$. Finally, NCL finds a local optimum of $\ell_{1+2}$ which retains good performance on task 1. See Appendix J for further mathematical details.

of previous and current tasks to the loss. However, to perform well in practice, weight regularization approaches have been found to require ad-hoc re-weighting of the prior term by several orders of magnitude [17, 29, 33]. We illustrate the shortcomings of weight regularization on a simple continual regression problem in Figure 1, where gradient descent on the Laplace posterior produces an indirect optimization path along which the first task is transiently forgotten as the second task is being learned. In addition, this can lead to catastrophic forgetting when the loss is non-convex (Figure 1B; green).

An alternative approach that has found recent success in a continual learning setting involves projection based methods which restrict parameter updates to a subspace that does not interfere with previous tasks [8, 39]. However, it is not immediately obvious how this projected subspace should be selected in a way that appropriately balances learning on previous and current tasks. Additionally, such projection based algorithms have fixed points that are minima of the current task, but not necessarily minima of the (negative) Bayesian posterior. This can lead to catastrophic forgetting in the limit of long training times (Figure 1; pink), unless the learning rate is exactly zero in directions that interfere with previous tasks.

To address these shortcomings, we introduce "Natural Continual Learning" (NCL) – an extension of the online Laplace approximation that also restricts parameter updates to directions which do not interfere strongly with previous tasks. In a Bayesian setting, we can conveniently express what is meant by such directions in terms of the prior precision matrix $\boldsymbol{\Lambda}$. In particular, 'flat' directions of the prior (low precision) correspond to directions that will not significantly affect the performance on previous tasks. Formally, we derive NCL as the solution of a trust region optimization problem. This involves maximizing the posterior loss $\mathcal{L}_k(\boldsymbol{\theta})$ within a region of radius $r$ centered around $\boldsymbol{\theta}$ with a distance metric of the form $d(\boldsymbol{\theta}, \boldsymbol{\theta} + \boldsymbol{\Delta}) = \sqrt{\boldsymbol{\Delta}^\top \boldsymbol{\Lambda}_{k-1} \boldsymbol{\Delta}/2}$ that takes into account the curvature of the prior via its precision matrix $\boldsymbol{\Lambda}_{k-1}$:

$$\boldsymbol{\Delta} = \arg\min_{\boldsymbol{\Delta}} \mathcal{L}_k(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}} \mathcal{L}_k(\boldsymbol{\theta})^\top \boldsymbol{\Delta} \quad \text{subject to} \quad \frac{1}{2} \boldsymbol{\Delta}^\top \boldsymbol{\Lambda}_{k-1} \boldsymbol{\Delta} \leq r^2, \tag{7}$$

where $\mathcal{L}_k(\boldsymbol{\theta} + \boldsymbol{\Delta}) \approx \mathcal{L}_k(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}} \mathcal{L}_k(\boldsymbol{\theta})^\top \boldsymbol{\Delta}$ is a first-order approximation to the updated Laplace objective. The solution to this subproblem is given by $\boldsymbol{\Delta} \propto \boldsymbol{\Lambda}_{k-1}^{-1} \nabla_{\boldsymbol{\theta}} \ell_k(\boldsymbol{\theta}) - (\boldsymbol{\theta} - \boldsymbol{\mu}_{k-1})$ (see Appendix A for a derivation), which gives rise to the NCL update rule

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \gamma \left[ \boldsymbol{\Lambda}_{k-1}^{-1} \nabla_{\boldsymbol{\theta}} \ell_k(\boldsymbol{\theta}) - (\boldsymbol{\theta} - \boldsymbol{\mu}_{k-1}) \right] \tag{8}$$

for a learning rate parameter $\gamma$ (which is implicitly a function of $r$ in Equation 8). To get some intuition for this learning rule, we note that $\boldsymbol{\Lambda}_{k-1}$ acts as a preconditioner for the first (likelihood) term which drives learning on the current task while encouraging parameter changes predominantly in directions that do not interfere with previous tasks. Meanwhile, the second term encourages $\boldsymbol{\theta}$ to

stay close to $\boldsymbol{\mu}_{k-1}$, the optimal parameters for the previous task. As we illustrate in Figure 1, this combines the desirable features of both Bayesian weight regularization and projection based methods. In particular, NCL shares the fixed points of the Bayesian posterior while also mitigating intermediate or complete forgetting of previous tasks by preconditioning with the prior covariance. Notably, if the loss landscape is non-convex (as it generally will be), NCL can converge to a different local optimum from standard weight regularization (Figure 1B).

**Implementation** In this work, we use a Kronecker-factored approximation to each Fisher matrix $\boldsymbol{F}_k$ in Equation 6 [24, 29], although we note that the general NCL framework can be applied with other approximations to $\boldsymbol{F}_k$ such as the diagonal approximation of Kirkpatrick et al. [17]. A major challenge in implementing NCL lies in the computation of $\boldsymbol{\Lambda}_{k-1}^{-1}$, which is generally intractable for large models. Even after making a Kronecker-factored approximation to $\boldsymbol{F}_k$ for each task $k$, it remains difficult to compute the inverse of a sum of $k$ Kronecker products (c.f. Equation 5). To address this challenge, we derived an efficient algorithm for making a Kronecker-factored approximation to $\boldsymbol{\Lambda}_k = \boldsymbol{F}_k + \boldsymbol{\Lambda}_{k-1} \approx \boldsymbol{L}_k \otimes \boldsymbol{R}_k$ when $\boldsymbol{\Lambda}_{k-1} = \boldsymbol{L}_{k-1} \otimes \boldsymbol{R}_{k-1}$ and $\boldsymbol{F}_k$ are also Kronecker products. This approximation minimizes the KL-divergence between $\mathcal{N}(\boldsymbol{\mu}_k, (\boldsymbol{L}_k \otimes \boldsymbol{R}_k)^{-1})$ and $\mathcal{N}(\boldsymbol{\mu}_k, (\boldsymbol{\Lambda}_{k-1} + \boldsymbol{F}_k)^{-1})$ (see Appendix C for details). The NCL algorithm is described in pseudocode in Appendix B together with additional implementation and computational details. Finally, while we have derived NCL with a Laplace approximation in this section for simplicity, it can similarly be applied in the variational continual learning framework of Nguyen et al. [25] (Appendix I).

### 2.3 Related work

As discussed in Section 2.1, our method is derived from prior work that relies on Bayesian inference to perform weight regularization for continual learning [12, 17, 25, 29]. However, we also take inspiration from the literature on natural gradient descent [1, 18] to introduce a preconditioner that encourages parameter updates primarily in flat directions of previously learned tasks (Appendix F).

Recent projection based methods [8, 39] have addressed the continual learning problem using an update rule of the form

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \gamma \boldsymbol{P}_L \nabla_{\boldsymbol{\theta}} \ell_k(\boldsymbol{\theta}) \boldsymbol{P}_R, \tag{9}$$

where $\boldsymbol{P}_L$ and $\boldsymbol{P}_R$ are projection matrices constructed from previous tasks which encourage parameter updates that do not interfere with performance on these tasks. Using Kronecker identities, we can rewrite Equation 9 as

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \gamma (\boldsymbol{P}_R \otimes \boldsymbol{P}_L) \nabla_{\boldsymbol{\theta}} \ell_k(\boldsymbol{\theta}). \tag{10}$$

This resembles the NCL update rule in Equation 8 where we identify $\boldsymbol{P}_R \otimes \boldsymbol{P}_L$ with the approximate inverse prior precision matrix used for gradient preconditioning in NCL $\boldsymbol{\Lambda}_{k-1}^{-1} = \boldsymbol{L}_{k-1}^{-1} \otimes \boldsymbol{R}_{k-1}^{-1}$. Indeed, we note that for a Kronecker-structured approximation to $\boldsymbol{F}_k$, the matrix $\boldsymbol{L}_{k-1}$ is the empirical covariance matrix of the network activations experienced during all tasks up to $k-1$ (3, 24, Appendix D) which is exactly the inverse of the projection matrix used in previous work [8, 39]. We thus see that NCL takes the form of recent projection based continual learning algorithms with two notable differences:

(i) NCL uses a right Kronecker factor $\boldsymbol{R}_{k-1}$ designed to approximate the posterior precision of previous tasks (Appendix D) while Duncker et al. [8] use the covariance of recurrent inputs and Zeng et al. [39] use the identity matrix $\boldsymbol{I}$. Notably, both of these choices of $\boldsymbol{R}_{k-1}$ still provide reasonable approximations to the prior Fisher matrix which can be used to motivate OWM and DOWM as projecting out steep directions of the prior (Appendix E).

(ii) NCL includes an additional regularization term $(\boldsymbol{\theta} - \boldsymbol{\mu}_{k-1})$ derived from the Bayesian posterior objective while Duncker et al. [8] and Zeng et al. [39] do not use such regularization. Importantly, this means that while NCL has a similar preconditioner and optimization path to these projection based methods, NCL has stationary points at the modes of the approximate Bayesian posterior while the stationary points of OWM and DOWM do not incorporate prior information from previous tasks (c.f. Figure 1).

It is also interesting to note that previous Bayesian continual learning algorithms include a hyperparameter $\lambda$ that scales the prior compared to the likelihood term for the current task [20]:

$$\mathcal{L}_k^{(\lambda)}(\boldsymbol{\theta}) = \log p(\mathcal{D}_k|\boldsymbol{\theta}) - \lambda(\boldsymbol{\theta} - \boldsymbol{\mu}_{k-1})^\top \boldsymbol{\Lambda}_{k-1}(\boldsymbol{\theta} - \boldsymbol{\mu}_{k-1}). \tag{11}$$

To minimize this loss and thus find a mode of the approximate posterior, it is common to employ pseudo-second-order stochastic gradient-based optimization algorithms such as Adam [16] that use their own gradient preconditioner based on an approximation to the Hessian of Equation 11. Interestingly, this Hessian is given by $\boldsymbol{H}_k = -H(\mathcal{D}_k, \boldsymbol{\theta}) - \lambda \boldsymbol{\Lambda}_{k-1}$, which in the limit of large $\lambda$ becomes increasingly similar to preconditioning with the prior precision as in NCL. Consistent with this, previous work using the online Laplace approximation has found that large values of $\lambda$ are generally required for good performance [17, 29, 33]. Recent work has also combined Bayesian continual learning with natural gradient descent [26, 31], and in this case a relatively high value of $\lambda = 100$ was similarly found to maximize performance [26].

# 3 Experiments and results

## 3.1 NCL in recurrent neural networks

As discussed in Section 1, we consider experiments in recurrent neural networks (RNNs), a setting that has recently proven challenging for continual learning [8, 9]. Here, we briefly describe the network dynamics and how the NCL algorithm (Section 2.2) is implemented.

**Network dynamics** The dynamics of the RNN used in this work can be described by the following equations:

$$\boldsymbol{h}_t = \boldsymbol{A}\boldsymbol{r}_{t-1} + \boldsymbol{B}\boldsymbol{x}_t + \boldsymbol{\xi}_t = \boldsymbol{W}\boldsymbol{z}_t + \boldsymbol{\xi}_t \tag{12}$$

$$\boldsymbol{y}_t \sim p(\boldsymbol{y}_t | \boldsymbol{C}\boldsymbol{r}_t) \tag{13}$$

where we define $\boldsymbol{r}_t = \phi(\boldsymbol{h}_t)$, $\boldsymbol{z}_t = (\boldsymbol{r}_{t-1}^\top, \boldsymbol{x}_t^\top)^\top$, $\boldsymbol{W} = (\boldsymbol{A}^\top, \boldsymbol{B}^\top)^\top$, and time is indexed by $t$. Here, $\boldsymbol{r} \in \mathbb{R}^{N_{rec} \times 1}$ are the network activations, $\boldsymbol{x} \in \mathbb{R}^{n_{in} \times 1}$ are the inputs, and $\boldsymbol{y} \in \mathbb{R}^{n_{out} \times 1}$ are the network outputs. The noise model $p(\boldsymbol{y}_t | \boldsymbol{C}\boldsymbol{r}_t)$ may be a Gaussian distribution for a regression task or a categorical distribution for a classification task, and $\phi(\boldsymbol{h})$ is a nonlinearity that is applied to $\boldsymbol{h}$ element-wise (in this work the ReLU function). The parameters of the RNN are given by $\boldsymbol{\theta} = (\boldsymbol{W}, \boldsymbol{C})$. The process noise $\{\boldsymbol{\xi}_t\}$ are zero-mean Gaussian random variables with covariance matrices $\boldsymbol{\Sigma}_t^{\boldsymbol{\xi}}$. In this model, the log-likelihood of observing a sequence of outputs $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T$ given inputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ and $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_T$ is given by

$$\ell(\boldsymbol{\theta}) = \log p_\theta(\{\boldsymbol{y}\} | \{\boldsymbol{x}\}, \{\boldsymbol{\xi}\}) = \log p(\{\boldsymbol{y}\} | \{\boldsymbol{C}\boldsymbol{r}\}), \tag{14}$$

**Kronecker-factored approximation to the FIMs** We use a Kronecker-factored approximation to the Fisher information matrix of $\ell(\boldsymbol{\theta})$ with respect to the parameters $\boldsymbol{C}$ and $\boldsymbol{W}$ (see Appendix D or section 3.4 of 23 for details). Defining $\overline{\boldsymbol{x}} := \partial \ell / \partial \mathrm{vec}(\boldsymbol{X})$, we approximate the FIMs of $\boldsymbol{C}$ and $\boldsymbol{W}$ as:

$$\boldsymbol{F}_{\boldsymbol{C}} = \mathbb{E}_{\{(\boldsymbol{\xi},\boldsymbol{x},\boldsymbol{y})\} \sim \mathcal{M}} \left[ \overline{\boldsymbol{c}}\, \overline{\boldsymbol{c}}^\top \right] \approx \mathbb{E}\left[T\right] \mathbb{E}_{\{(\boldsymbol{\xi},\boldsymbol{x},\boldsymbol{y})\} \sim \mathcal{M}} \left[ \boldsymbol{r}\boldsymbol{r}^\top \right] \otimes \mathbb{E}_{\{(\boldsymbol{\xi},\boldsymbol{x},\boldsymbol{y})\} \sim \mathcal{M}} \left[ \overline{\boldsymbol{y}}\, \overline{\boldsymbol{y}}^\top \right] \tag{15}$$

$$\boldsymbol{F}_{\boldsymbol{W}} = \mathbb{E}_{\{(\boldsymbol{\xi},\boldsymbol{x},\boldsymbol{y})\} \sim \mathcal{M}} \left[ \overline{\boldsymbol{w}}\, \overline{\boldsymbol{w}}^\top \right] \approx \mathbb{E}\left[T\right] \mathbb{E}_{\{(\boldsymbol{\xi},\boldsymbol{x},\boldsymbol{y})\} \sim \mathcal{M}} \left[ \boldsymbol{z}\boldsymbol{z}^\top \right] \otimes \mathbb{E}_{\{(\boldsymbol{\xi},\boldsymbol{x},\boldsymbol{y})\} \sim \mathcal{M}} \left[ \overline{\boldsymbol{h}}\, \overline{\boldsymbol{h}}^\top \right], \tag{16}$$

where $\mathbb{E}\left[T\right]$ is the expected length of each input-output sequence, and we take $\{(\boldsymbol{\xi}, \boldsymbol{x}, \boldsymbol{y})\} \sim \mathcal{M}$ to mean that $\{\boldsymbol{\xi}\}$, $\{\boldsymbol{x}\}$ and $\{\boldsymbol{y}\}$ are drawn from the model distribution defined in Equation 12.

## 3.2 Stimulus-response tasks

In this section, we compare different methods for continual learning on a set of neuroscience inspired 'stimulus-response' (SR) tasks (37; details in Appendix G).

Following previous work, we first considered RNNs with 256 units [8, 36]. While NCL, OWM and DOWM all managed to learn the six tasks without catastrophic forgetting (Figure 2A–C), we found that NCL achieved superior average performance across all tasks after training (Figure 3A). We then compared NCL, OWM, and DOWM to KFAC, an online Laplace algorithm that uses Adam [16] to optimize the objective in Equation 4 with a Kronecker-factored approximation to the precision matrix in Equation 5 (Section 2.1; 29). Consistent with the results shown in Duncker et al. [8], we found that NCL, OWM, and DOWM outperformed KFAC (Figure 2D; see also 8 for a comparison of DOWM and EWC). We note that NCL and KFAC optimize the same objective function (Equation 4) and

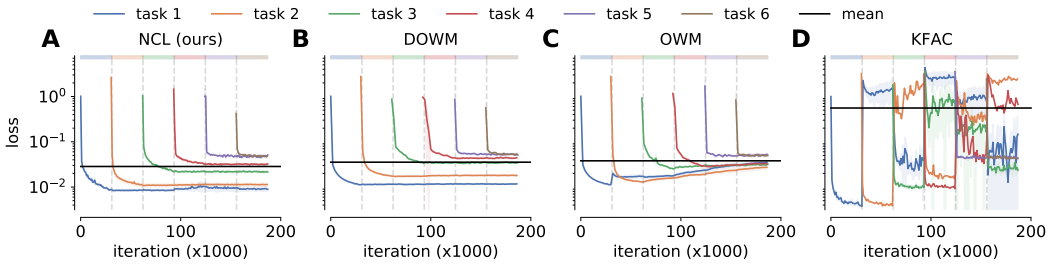


Figure 2: **Continual learning on stimulus-response tasks.** Evolution of the loss during training for each of the six stimulus-response tasks for NCL (A), DOWM (B), OWM (C), and KFAC with $\lambda = 1$ (D). Coloured lines and shadings indicate mean and stdev for each task across 5 random seeds. Black lines indicate the mean loss over all tasks at the end of training. Task order matches that of Duncker et al. [8]. Note that the earlier tasks are also 'easier' in a non-continual setting [37].

approximate the posterior precision matrix in the same way, but differ in the way they precondition the gradient of the objective. Our results thus demonstrate empirically that the choice of optimization algorithm is important to prevent forgetting, consistent with the intuition provided by Figure 1.

In previous work, poor performance with weight regularization approaches such as EWC and KFAC has been mitigated by introducing a hyperparameter $\lambda$ that increases the importance of the prior term compared to a standard Bayesian treatment (Equation 11; 17, 20, 29). We confirmed this here by performing a grid search over $\lambda$, which showed that KFAC with $\lambda \in [100, 1000]$ could perform almost as well as NCL (Section H.1; Figure 3A). We hypothesize that the good performance provided by high $\lambda$ is partly due to the approximate second order nature of Adam which, together with the relative increase in the prior term compared to the data term, leads to preconditioning with a matrix resembling the prior $\mathbf{\Lambda}_{k-1}$ (Section 2.3). In support of this hypothesis, we found that the KL divergence between the Adam preconditioner and the approximate prior precision $\mathbf{\Lambda}_{k-1}$ decreased with increasing $\lambda$, and that the performance of KFAC with Adam could also be rescued by increasing $\lambda$ only when computing the preconditioner while retaining $\lambda = 1$ when computing the gradients (Section H.1).

Duncker et al. have previously shown that the dynamics of RNNs trained to perform the SR tasks of Figure 2 are low dimensional, suggesting that 256 units may not be needed to learn the tasks in a continual fashion. We therefore reduced the network size to 50 units to further challenge the continual learning algorithms. We found that NCL outperformed both DOWM and OWM, suggesting that a good approximation to the Fisher matrix is particularly important in more challenging settings (Figure 3B; Section H.2).

### 3.3 Stroke MNIST

Another way to challenge the CL algorithms further is to increase the number of tasks. We thus considered an augmented version of the stroke MNIST dataset (SMNIST; 6). The original dataset consists of the MNIST digits transformed into pen strokes with the direction of the stroke at each time point provided as inputs to the network. Similar to Ehret et al. [9], we constructed a continual learning problem by considering consecutive binary classification tasks inspired by the canonical split MNIST task set. We further increased the number of tasks by including a set of extra digits where the x and y dimensions have been swapped in the input stroke data, and another set where both the x and y dimensions have changed sign. We also added high-variance noise to the inputs to increase the task difficulty. This gave rise to a total of 15 binary classification tasks, each with unique digits not used in other tasks, which we sought to learn in a continual fashion using an RNN with 30 recurrent units (see Appendix G for details).

As for the SR task set in Section 3.2, we found that NCL outperformed previous projection based methods (Figure 3C). We again found that weight regularization with a KFAC approximation performed poorly with $\lambda = 1$, and that this poor performance could be partially rescued by optimizing over $\lambda$ (Figure 3C). To investigate how the difference in performance between NCL and DOWM was affected by their different approximations to the Fisher matrix (Appendix E), we implemented NCL using the DOWM projection matrices as an alternative approximation to the inverse Fisher matrix. We refer to this method as Laplace-DOWM. We then considered how the performance on each task at

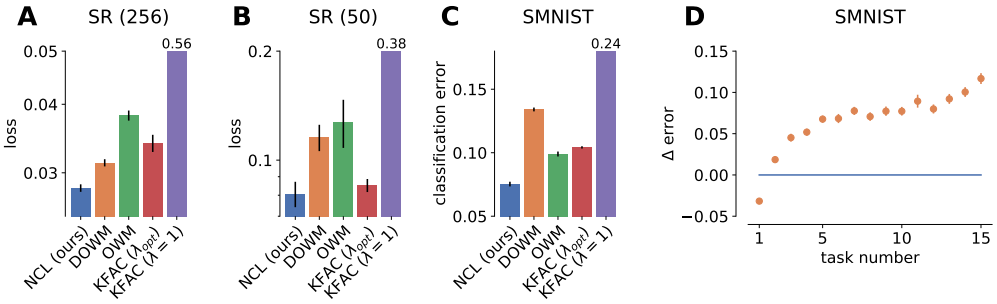


Figure 3: **Performance on SR and SMNIST tasks. (A)** Mean loss of NCL, DOWM, OWM, KFAC (optimal $\lambda$), and KFAC ($\lambda = 1$) across stimulus-response tasks for RNNs with 256 units. Here and in (B) and (C), KFAC with $\lambda = 1$ failed catastrophically and its performance is indicated in text as it does not fit on the axes. **(B)** As in (A), now for networks with 50 units. Error bars in (A) and (B) indicate standard error (s.e.m) across 5 random seeds. **(C)** Mean classification error across SMNIST tasks for networks with 30 hidden units. **(D)** Difference between the mean classification error of Laplace-DOWM and NCL as a function of task number. Error bars in (C) and (D) indicate s.e.m. across 100 random task permutations.

the end of training depended on task number, averaged over different task permutations (Figure 3D). We found that while Laplace-DOWM outperformed NCL on the first task, this method generally performed worse on subsequent tasks. Notably, Laplace-DOWM exhibited a near-monotonic decrease in relative performance with task number which is consistent with the intuition that DOWM overestimates the dimensionality of the parameter subspace that matters for previous tasks (Appendix E). In contrast, although neural circuits are known to use orthogonal subspaces in different contexts, there is no general sense that learning more tasks in the past should systematically hinder learning in future contexts for biological agents.

### 3.4 Dissecting the dynamics of networks trained on the SMNIST task set

To further investigate how the RNNs solve the continual learning problems and how this relates to the neuroscience literature, we dissected the dynamics of networks trained on the SMNIST task set using the NCL algorithm (see Section H.4 for an equivalent analysis with DOWM). To do this, we analyzed latent representations of the RNN activity trajectories, as is commonly done to study the collective dynamics of artificial and biological networks [10, 13, 22, 38]. We considered two consecutive classification tasks, namely classifying 4's vs 5's ($k = 2$) and classifying 1's vs 7's ($k = 3$). For each of these tasks, we trained a factor analysis model right after the task was learned, using network activity collected while presenting 50 examples of each of the two input digits associated with the task. We then tracked the network responses to the same set of stimuli at various stages of learning, both before and after the task in question was acquired, using the trained factor analysis model to visualize low-dimensional summaries of the dynamics (Figure 4).

Consistent with the network having successfully learned to solve these two tasks, we found that latent trajectories diverged over time for the two types of inputs in each task. Critically, these diverging dynamics only emerged after the task was learned, and remained highly stable thereafter (Figure 4). The stability of the task-associated representations is consistent with recent work in the neuroscience literature showing that, in a primate reaching task, latent neural trajectories remain stable after learning [10]. Since here we have access to the activity of all neurons throughout the task, we proceeded to quantify the source of this stability at the level of single units. The stability of such single-neuron dynamics after learning has recently been a topic of contention in biological circuits [5, 21]. In the RNNs, we found that the single-unit representations of a given digit changed during learning of the task involving that digit but stabilized after learning, consistent with work in several distinct biological circuits [4, 7, 11, 14, 28].

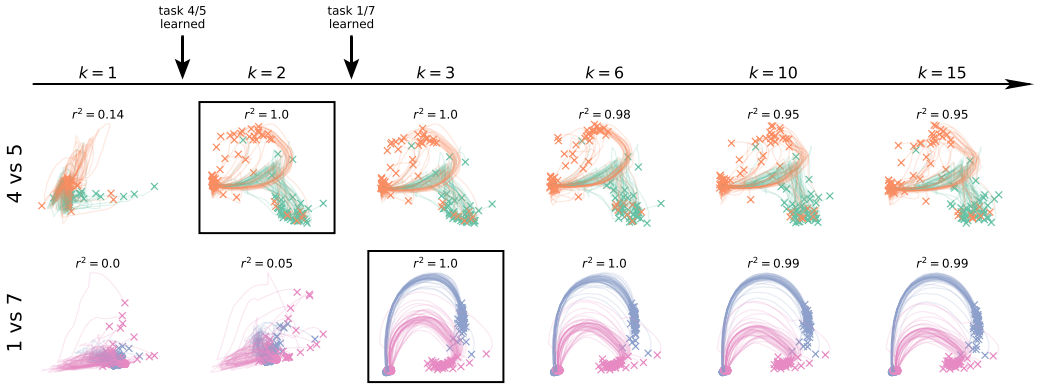


Figure 4: **Latent dynamics during SMNIST.** We considered two example tasks, 4 vs 5 (top) and 1 vs 7 (bottom). For each task, we simulated the response of a network trained by NCL to 100 digits drawn from that task distribution at different times during learning. We then fitted a factor analysis model for each example task to the response of the network right after the correponding task had been learned (squares; $k = 2$ and $k = 3$ respectively). We used this model to project the responses at different times during learning into a common latent space for each example task. For both example tasks, the network initially exhibited variable dynamics with no clear separation of inputs and subsequently acquired stable dynamics after learning to solve the task. The $r^2$ values above each plot indicate the similarity of neural population activity with that collected immediately after learning the corresponding task, quantified across all neurons (not just the 2D projection).

## 4 Discussion

In summary, we have developed a new framework for continual learning based on approximate Bayesian inference combined with trust-region optimization. We showed that this framework encompasses recent projection based methods and found that it performs better than naive weight regularization in a recurrent neural network setting which has previously been shown to be challenging for various continual learning algorithms [8, 9]. Furthermore, we showed that our principled probabilistic approach outperforms previous projection based methods [8, 39], in particular when the number of tasks and their complexity challenges the network's capacity. Finally, we analyzed the dynamics of the learned networks in a sequential binary classification problem where we found that the latent dynamics adapt to each new task. We also found that the task-associated dynamics were subsequently conserved during further learning, consistent with experimental reports of stable neural representations [7, 10]. Importantly, our results suggest that preconditioning with the prior covariance can lead to improved performance over existing continual learning algorithms. In future work, it will therefore be interesting to use NCL with other weight regularization approaches such as EWC [17], and to extend its use to feedforward neural networks. Finally, a separate branch of continual learning utilizes generative replay or functional regularization based on previous data and models [19, 27, 30, 33]. While our work has focused on weight regularization, such regularization and replay is not mutually exclusive. Instead, these two approaches have been found to further improve robustness to catastrophic forgetting when combined [25, 32].

**Impact** As machine learning systems become more prominent in our everyday lives, they increasingly need to be robust to changing data distributions and dynamic task specifications. Much work has therefore gone into the development of methods for continual learning in the machine learning community. However, with the increasing prevalence of practical algorithms for continual learning, it becomes increasingly important that we understand how and why these algorithms work – insights that can also help us understand when they might fail. In this work, we have therefore attempted to shed light on the relationship between recent methods for continual learning as well as developing a new algorithm with a principled probabilistic interpretation that makes explicit its underlying assumptions and shortcomings. Taken together, we hope that this work will help improve our understanding of methods for continual learning while also providing an avenue for further research to increase the reliability and robustness of future continual learning algorithms.

# References


[1] Amari, S.-I. (1998). Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276.

[2] Ames, K. C. and Churchland, M. M. (2019). Motor cortex signals for each arm are mixed across hemispheres and neurons yet partitioned within the population response. *Elife*, 8:e46159.

[3] Bernacchia, A., Lengyel, M., and Hennequin, G. (2018). Exact natural gradient in deep linear networks and its application to the nonlinear case. *Advances in Neural Information Processing Systems*, 31:5941–5950.

[4] Chestek, C. A., Batista, A. P., Santhanam, G., Byron, M. Y., Afshar, A., Cunningham, J. P., Gilja, V., Ryu, S. I., Churchland, M. M., and Shenoy, K. V. (2007). Single-neuron stability during repeated reaching in macaque premotor cortex. *Journal of Neuroscience*, 27(40):10742–10750.

[5] Clopath, C., Bonhoeffer, T., Hübener, M., and Rose, T. (2017). Variance and invariance of neuronal long-term representations. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 372(1715):20160161.

[6] de Jong, E. D. (2016). Incremental sequence learning. *arXiv preprint arXiv:1611.03068*.

[7] Dhawale, A. K., Poddar, R., Wolff, S. B., Normand, V. A., Kopelowitz, E., and Ölveczky, B. P. (2017). Automated long-term recording and analysis of neural activity in behaving animals. *Elife*, 6:e27702.

[8] Duncker, L., Driscoll, L., Shenoy, K. V., Sahani, M., and Sussillo, D. (2020). Organizing recurrent network dynamics by task-computation to enable continual learning. *Advances in Neural Information Processing Systems*, 33.

[9] Ehret, B., Henning, C., Cervera, M. R., Meulemans, A., von Oswald, J., and Grewe, B. F. (2020). Continual learning in recurrent neural networks with hypernetworks. *arXiv preprint arXiv:2006.12109*.

[10] Gallego, J. A., Perich, M. G., Chowdhury, R. H., Solla, S. A., and Miller, L. E. (2020). Long-term stability of cortical population dynamics underlying consistent behavior. *Nature neuroscience*, 23(2):260–270.

[11] Ganguly, K. and Carmena, J. M. (2009). Emergence of a stable cortical map for neuroprosthetic control. *PLoS Biol*, 7(7):e1000153.

[12] Huszár, F. (2017). On quadratic penalties in elastic weight consolidation. *arXiv preprint arXiv:1712.03847*.

[13] Jensen, K. T., Kao, T.-C., Tripodi, M., and Hennequin, G. (2020). Manifold gplvms for discovering non-euclidean latent structure in neural data. *arXiv preprint arXiv:2006.07429*.

[14] Katlowitz, K. A., Picardo, M. A., and Long, M. A. (2018). Stable sequential activity underlying the maintenance of a precisely executed skilled behavior. *Neuron*, 98(6):1133–1140.

[15] Kaufman, M. T., Churchland, M. M., Ryu, S. I., and Shenoy, K. V. (2014). Cortical activity in the null space: permitting preparation without movement. *Nature neuroscience*, 17(3):440–448.

[16] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[17] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

[18] Kunstner, F., Balles, L., and Hennig, P. (2019). Limitations of the empirical fisher approximation for natural gradient descent. *arXiv preprint arXiv:1905.12558*.

[19] Li, Z. and Hoiem, D. (2017). Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947.

[20] Loo, N., Swaroop, S., and Turner, R. E. (2020). Generalized variational continual learning. *arXiv preprint arXiv:2011.12328*.

[21] Lütcke, H., Margolis, D. J., and Helmchen, F. (2013). Steady or changing? long-term monitoring of neuronal population activity. *Trends in neurosciences*, 36(7):375–384.

[22] Mante, V., Sussillo, D., Shenoy, K. V., and Newsome, W. T. (2013). Context-dependent computation by recurrent dynamics in prefrontal cortex. *nature*, 503(7474):78–84.

[23] Martens, J., Ba, J., and Johnson, M. (2018). Kronecker-factored curvature approximations for recurrent neural networks. In *International Conference on Learning Representations*.

[24] Martens, J. and Grosse, R. (2015). Optimizing neural networks with kronecker-factored approximate curvature. In *ICML*, pages 2408–2417.

[25] Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. (2017). Variational continual learning. *arXiv preprint arXiv:1710.10628*.

[26] Osawa, K., Swaroop, S., Jain, A., Eschenhagen, R., Turner, R. E., Yokota, R., and Khan, M. E. (2019). Practical deep learning with bayesian principles. *arXiv preprint arXiv:1906.02506*.

[27] Pan, P., Swaroop, S., Immer, A., Eschenhagen, R., Turner, R. E., and Khan, M. E. (2020). Continual deep learning by functional regularisation of memorable past. *arXiv preprint arXiv:2004.14070*.

[28] Peters, A. J., Chen, S. X., and Komiyama, T. (2014). Emergence of reproducible spatiotemporal activity during motor learning. *Nature*, 510(7504):263–267.

[29] Ritter, H., Botev, A., and Barber, D. (2018). Online structured laplace approximations for overcoming catastrophic forgetting. *arXiv preprint arXiv:1805.07810*.

[30] Shin, H., Lee, J. K., Kim, J., and Kim, J. (2017). Continual learning with deep generative replay. *arXiv preprint arXiv:1705.08690*.

[31] Tseran, H., Khan, M. E., Harada, T., and Bui, T. D. (2018). Natural variational continual learning. In *Continual Learning Workshop@ NeurIPS*, volume 2.

[32] van de Ven, G. M., Siegelmann, H. T., and Tolias, A. S. (2020). Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, 11(1):1–14.

[33] Van de Ven, G. M. and Tolias, A. S. (2018). Generative replay with feedback connections as a general strategy for continual learning. *arXiv preprint arXiv:1809.10635*.

[34] von Oswald, J., Henning, C., Sacramento, J., and Grewe, B. F. (2019). Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*.

[35] Xu, T., Yu, X., Perlik, A. J., Tobin, W. F., Zweig, J. A., Tennant, K., Jones, T., and Zuo, Y. (2009). Rapid formation and selective stabilization of synapses for enduring motor memories. *Nature*, 462(7275):915–919.

[36] Yang, G., Pan, F., and Gan, W.-B. (2009). Stably maintained dendritic spines are associated with lifelong memories. *Nature*, 462(7275):920–924.

[37] Yang, G. R., Joglekar, M. R., Song, H. F., Newsome, W. T., and Wang, X.-J. (2019). Task representations in neural networks trained to perform many cognitive tasks. *Nature neuroscience*, 22(2):297–306.

[38] Yu, B. M., Cunningham, J. P., Santhanam, G., Ryu, S. I., Shenoy, K. V., and Sahani, M. (2009). Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. *Journal of neurophysiology*, 102(1):614–635.

[39] Zeng, G., Chen, Y., Cui, B., and Yu, S. (2019). Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372.

# Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The code used for this work has been included in the supplementary material and will be made freely available on github following the anonymous review period.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] The stimulus-response tasks were adapted from Yang et al. [37] and the SMNIST tasks from de Jong [6] (released under a Creative Commons Attribution-Share Alike 3.0 license).
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] An implementation of NCL in jax is included in the supplementary material and will be made freely available on github following the anonymous review period.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]