ONG: Orthogonal Natural Gradient Descent

Yajat Yadav
UC Berkeley
yajatyadav@berkeley.edu

Patrick Mendoza
UC Berkeley
patmendoza6745@berkeley.edu

Jathin Korrapati
UC Berkeley
jkorr@berkeley.edu

Abstract

Orthogonal Gradient Descent (OGD) has emerged as a powerful method for continual learning. However, its Euclidean projections do not leverage the underlying information-geometric structure of the problem, which can lead to suboptimal convergence in learning tasks. To address this, we propose incorporating the natural gradient into OGD and present **ONG (Orthogonal Natural Gradient Descent)**. ONG preconditions each new task-specific gradient with an efficient EKFAC approximation of the inverse Fisher information matrix, yielding updates that follow the steepest descent direction under a Riemannian metric. To preserve performance on previously learned tasks, ONG projects these natural gradients onto the orthogonal complement of prior tasks' natural gradients. We provide an initial theoretical justification for this procedure, introduce the Orthogonal Natural Gradient Descent (ONG) algorithm, and present preliminary results on the Permuted and Rotated MNIST benchmarks. Our preliminary results, however, indicate that a naive combination of natural gradients and orthogonal projections has potential issues. This finding has motivated continued future work focused on robustly reconciling these geometric perspectives to develop a continual learning method, establishing a more rigorous theoretical foundation with formal convergence guarantees, and extending empirical validation to large-scale continual learning benchmarks. The anonymized version of our code can be found as the zip file here.

1 Introduction

Continual learning, the process of training a single model on a sequence of tasks without forgetting previously learned tasks, is a major challenge in deep learning. Naive fine-tuning leads to catastrophic forgetting, where earlier tasks' performance collapses as networks shift their weights to accommodate the new task, forgetting previously learned tasks [1, 2]. In deep learning, a promising training-time algorithm to mitigate this issue is Orthogonal Gradient Descent (OGD) [3]. OGD works by projecting each task gradient to an orthogonal subspace spanned by previous tasks' gradients. By subtracting off this projection before taking the gradient step, we ensure that the model retains performance on previously seen tasks. Another idea in optimization is the natural gradient. Natural gradients represent the steepest descent direction with respect to the underlying geometry of the parameter space. Iteratively following the natural gradient yields the natural gradient descent algorithm [4]. In many applications, Natural Gradient Descent requires far fewer iterations to converge than standard gradient descent. It is also invariant to any smooth and invertible reparameterization of the model, as compared to gradient descent, which is parametrization dependent [5]. Natural Gradient Descent works by modifying the gradient update rule to account for the "information geometry" of the parameter space. In the context of probabilistic models, this involves preconditioning the gradient in the update rule with the inverse Fisher information matrix (since the Riemannian metric structure of the parameter space is described by the Fisher information [4]). Natural Gradient Descent methods have been used in several applications, such as reinforcement learning [6] and variational inference [7].

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Non-Euclidean Foundation Models: Advancing AI Beyond Euclidean Frameworks.

In this work, we present **Orthogonal Natural Gradient Descent (ONG)**, a novel algorithm for continual learning that integrates natural gradients within the Orthogonal Gradient Descent (OGD) framework. ONG is designed to harness the complementary strengths of these methods: the explicit forgetting prevention of OGD and the efficient, geometrically-informed updates of natural gradient optimization. We formalize the ONG algorithm, establish preliminary theoretical guarantees for its performance, and provide an efficient implementation. However, our empirical results on standard continual learning benchmarks suggests that naively combining natural gradients with orthogonal projections not only doesn't mitigate catastrophic forgetting, but also seems to worsen the performance of vanilla OGD. This suggests some fundamental geometric incompatibility between Euclidean projections and the Fisher-metric space, and has inspired us to continue future work on rigorously developing and justifying the correct way to develop geometry-consistent projection operators. We are currently working on investigating the exact failure modes of our method to better understand why this naive combination of two ideas starts to degrade performance. In addition, we are simultaneously working on extending our evaluation criteria to incorporate more challenging and realistic continual learning settings.

2 Related Work

Continual Learning Methods Continual learning methods broadly fall into three categories: regularization-based, replay-based, and dynamic architecture methods. Regularization-based methods add task-specific penalties to the loss to preserve old knowledge. Notable works include Elastic Weight Consolidation [8], which uses the Fisher information at a task's optimum as a quadratic penalty on parameter changes, and Synaptic Intelligence (SI) [9], which integrates a Hebbian-like importance of each weight over time. Replay-based methods mitigate forgetting by rehearsing data from previous tasks during new-task training. This can be done by either storing real data or generating synthetic examples that approximate prior distributions. For example, deep Generative Replay (DGR) methods [10] train a generative model to mimic old data distributions. During new task training, the generative model samples pseudo-data from prior tasks, allowing the learner to "replay" previous experiences without storing real examples. Finally, dynamic architecture methods seek to address catastrophic forgetting by modifying the model structure over time, often by allocating task-specific sub-modules or expanding capacity to accommodate new tasks. For instance, Progressive Neural Networks (PNNs) [11] allocate a new column for each task and connect it to previous columns via lateral connections. This guarantees no forgetting but scales poorly with the number of tasks.

A notable continual learning method, which inspires the foundation of our work, is Orthogonal Gradient Descent (OGD) [3]. OGD mitigates forgetting by projecting incoming task gradients onto the orthogonal complement of the subspace spanned by previous tasks' observed gradients. Follow-up work by Bennani et al. [12] offers theoretical guarantees for OGD in the neural tangent kernel (NTK) regime, demonstrating that under mild assumptions, OGD can preserve task performance indefinitely.

Natural Gradient Methods Natural Gradient Descent [4] generalizes vanilla gradient descent by taking steps that respect the geometry of the parameter space. Instead of updating in the Euclidean parameter-space, natural gradients perform optimization in the space of probability distributions, where the Fisher information matrix (FIM) serves as a Riemannian metric. In probabilistic models and Bayesian neural networks, natural gradient descent yields faster convergence and more stable updates by accounting for parameter sensitivity [13, 5].

3 Preliminaries

3.1 Continual Learning

In continual learning, a model encounters tasks $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, ...$ sequentially, each of which is its own supervised learning problem. Once the data for task $\mathcal{T}_k$ has been utilized for training, it is usually discarded (each example is seen once). The challenge lies in updating the model so that it performs well on the current task without sacrificing its ability to solve all previously learned tasks [14]. Formally, we assume that for each task $\mathcal{T}_k$, the data $\{(\boldsymbol{x}_{k,i}, y_{k,i})\}_{i=1}^{n_k}$ are drawn i.i.d. from some distribution $\mathcal{D}_k$. We denote the model parameters after t iterations on task k by $f_k^{(t)}$, and then use f_k^* to refer to the final parameters of the trained model after seeing k tasks.

3.2 Probabilistic Setup

When we train a neural network for classification, we implicitly fit a parametric family of probability distributions over the class labels. More formally, a multi-layer perceptron with softmax outputs defines

$$p\,(y \mid \boldsymbol{x}; \boldsymbol{w}) = \frac{\exp(f_y(\boldsymbol{x}; \boldsymbol{w}))}{\sum_{\mathrm{c}} \exp(f_{\mathrm{c}}(\boldsymbol{x}; \boldsymbol{w}))} \tag{1}$$

where $f_{\mathrm{c}}(\boldsymbol{x}; \boldsymbol{w})$ is the logit for class c. Our objective is to maximize the log-likelihood (equivalent to minimizing cross-entropy) so the learning problem lives on the manifold of all these predictive distributions. This is what motivates the use of the Fisher information matrix [4]:

$$\mathbf{F}(\mathbf{w}) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D},\; y\sim p(y \mid \mathbf{x};\mathbf{w})} \left[\nabla_{\mathbf{w}} \log p\,(y \mid \mathbf{x}; \mathbf{w}) \cdot \nabla_{\mathbf{w}} \log p\,(y \mid \mathbf{x}; \mathbf{w})^{\top}\right] \tag{2}$$

Amari [4] demonstrates that a Riemannian structure is given to the parameter space of multilayer networks by the Fisher information matrix. Intuitively, $\mathbf{F}(\mathbf{w})$ measures the sensitivity of the model's output distribution to small changes in each parameter direction around $\mathbf{w}$. As $\mathbf{F}(\mathbf{w})$ is the Hessian of the KL divergence between the distribution parametrized by $\mathbf{w}$ and some fixed distribution, directions of larger Fisher curvature correspond to larger changes in the output distribution. This geometric structure on our parameter space motivates the natural gradient

$$\delta(\boldsymbol{w}) = \boldsymbol{F}^{-1}(\boldsymbol{w}) \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}) \tag{3}$$

as being the steepest-descent direction of loss function $\mathcal{L}(\boldsymbol{w})$ while also constraining the KL-divergence between output distributions. The preconditioning by the inverse Fisher matrix provides us with optimization steps that are invariant to model reparameterization, which can be useful in tasks such as classification [5].

3.3 Orthogonal Gradient Descent

In continual learning, Orthogonal Gradient Descent (OGD) is a method that aims to prevent forgetting by projecting each new task gradient onto a subspace spanned by previous task gradients, in order to retain performance on previously seen tasks. We begin by restating the original algorithm and then summarize its key theoretical guarantees.

Algorithm 1 Orthogonal Gradient Descent

```
 1: Input: Task sequence $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \ldots$, learning rate $\eta$
 2: Output: The optimal parameter $\boldsymbol{w}$
 3: Initialize $S \leftarrow \{\}$; $\boldsymbol{w} \leftarrow \boldsymbol{w}_0$
 4: for Task ID $k = 1, 2, 3, \ldots$ do
 5:     repeat
 6:         $\boldsymbol{g} \leftarrow$ Stochastic/Batch Gradient for $\mathcal{T}_k$ at $\boldsymbol{w}$
 7:         $\tilde{\boldsymbol{g}} = \boldsymbol{g} - \sum_{\boldsymbol{v}\in S} \mathrm{proj}_{\boldsymbol{v}}(\boldsymbol{g})$
 8:         $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta\tilde{\boldsymbol{g}}$
 9:     until convergence
10:     for $(\boldsymbol{x}, y) \in \mathcal{T}_k$ and $k \in [1, c]$ such that $y_k = 1$ do
11:         $\boldsymbol{u} \leftarrow \nabla f_k(\boldsymbol{x}; \boldsymbol{w}) - \sum_{\boldsymbol{v}\in S} \mathrm{proj}_{\boldsymbol{v}}\left(\nabla f_k(\boldsymbol{x}; \boldsymbol{w})\right)$
12:         $S \leftarrow S \cup \{\boldsymbol{u}\}$
13:     end for
14: end for
```

Algorithm 1 (shown above) is the original OGD algorithm [3]. To justify the use of orthogonal gradient descent, it has been shown that every model update $\tilde{\boldsymbol{g}}$ remains a valid descent direction and that the generalization gap can be bounded just as tightly as it is for SGD [12]. The forgetting bound is also tightly controlled by a factor that can limit how much any one past task's loss can rise. Together, these results show that OGD is not just empirically effective, but also comes with provable convergence, bounded generalization error, and guaranteed forgetting control [3]. All these aspects make it theoretically sound and appealing for continual learning and our applications with the natural gradient.

4 Orthogonal Natural Gradient Descent

We now describe our initial attempt to combine the core ideas of OGD and natural gradients to form a new continual learning algorithm ONG: Orthogonal Natural Gradient Descent. This algorithm aims to combine the parameter space invariant updates of natural gradients with the interference-free projections of orthogonal gradient descent (OGD). The key idea is partitioned into several steps.

4.1 Natural Preconditioning

For each new task gradient, $\boldsymbol{g} = \nabla_{\boldsymbol{w}} \mathcal{L}_k(\boldsymbol{w})$, we modify this gradient to instead be $\boldsymbol{g} = \widehat{\mathbf{F}}(\mathbf{w})^{-1} \nabla_{\boldsymbol{w}} \mathcal{L}_k(\boldsymbol{w})$ where $\widehat{\mathbf{F}}(\mathbf{w})$ is an approximation of the Fisher information matrix for the current set of model parameters. This step moves the parameters $\boldsymbol{w}$ along the steepest direction on the manifold of the predictive distributions, which minimizes the KL-divergence from the conditional distribution parameterized by the previous model and ensures reparameterization invariance. The remaining steps from the previous algorithm remain the same. We still maintain an orthogonal basis S that spans the space of previous task gradients and project $\boldsymbol{g}$ into the orthogonal complement of span(S):

$$\tilde{\boldsymbol{g}} = \boldsymbol{g} - \sum_{\boldsymbol{v} \in S} \text{proj}_{\boldsymbol{v}}(\boldsymbol{g}) \tag{4}$$

We perform the same parameter update $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \tilde{\boldsymbol{g}}$, and, analogous to OGD, once task T_k converges, we extract each logitwise gradient $\nabla f_k(\boldsymbol{x}; \boldsymbol{w})$, precondition this gradient, project it onto the orthogonal complement of the subspace spanned by all previously added logitwise gradients, and denote the result by $\boldsymbol{u}$. We add $\boldsymbol{u}$ to our set S, maintaining S as an orthogonal basis of sensitive directions. Thus, the set S thus spans the subspace of all previously seen *natural* gradients.

4.2 OGD-Plus (OGD+)

OGD-Plus (OGD+) is an algorithm that enhances vanilla OGD by storing the feature maps with respect to samples from previous tasks, as well as the feature maps with respect to samples from the current task [12]. These extra samples are saved in a dedicated memory, the sample memory. The authors motivated this change in order to make OGD more robust to the NTK (Neural Tangent Kernel) variation phenomena in their experiments. The changes from OGD to OGD+ can be found in blue in the below algorithm.

Algorithm 2 ONG / ONG+

```
1: Input: Task sequence $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \ldots$, learning rate $\eta$
2: Output: The optimal parameter $\boldsymbol{w}$
3: Initialize $S \leftarrow \{\}$; $S_D \leftarrow \{\}$; $\boldsymbol{w} \leftarrow \boldsymbol{w}_0$
4: for Task ID $k = 1, 2, 3, \ldots$ do
5:     repeat
6:         $\boldsymbol{g} \leftarrow$ Stochastic/Batch Gradient for $\mathcal{T}_k$ at $\boldsymbol{w}$
7:         $\boldsymbol{g} \leftarrow \boldsymbol{F}^{-1}\boldsymbol{g}$
8:         $\tilde{\boldsymbol{g}} = \boldsymbol{g} - \sum_{\boldsymbol{v} \in S} \text{proj}_{\boldsymbol{v}}(\boldsymbol{g})$
9:         $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \tilde{\boldsymbol{g}}$
10:    until convergence
11:    Sample $H \subset S_D$
12:    for $(\boldsymbol{x}, y) \in \mathcal{T}_k \cup H$ and $k \in [1, c]$ such that $y_k = 1$ do
13:        $\boldsymbol{g} \leftarrow \boldsymbol{F}^{-1} \nabla f_k(\boldsymbol{x}; \boldsymbol{w})$
14:        $\boldsymbol{u} \leftarrow \boldsymbol{g} - \sum_{\boldsymbol{v} \in S} \text{proj}_{\boldsymbol{v}}(\boldsymbol{g})$
15:        $S \leftarrow S \cup \{\boldsymbol{u}\}$
16:    end for
17:    Sample $D \subset \mathcal{T}_k$
18:    Update $S_D \leftarrow S_D \cup D$
19: end for
```

4.3 ONG and ONG+

In this paper, we present ONG and ONG+, extensions of OGD and OGD+ using natural gradients, respectively. Our changes are in red in Algorithm 2: we precondition gradients using the Fisher information matrix both when updating model parameters and when storing task gradients in memory (the span of this set is what future gradients are ultimately projected onto).

4.4 Theoretical Guarantees

This section focuses on studying the ONG algorithm (aka Algorithm 2 without the blue text) to obtain some theoretical results. Analogous to Lemma 3.1 from the original OGD paper [3], we present a proof that the preconditioned, projection-subtracted gradient $\tilde{\boldsymbol{g}}$ is still a descent direction with respect to the Fisher metric.

As described in section 3.1, in the context of probabilistic models, the Riemannian metric tensor is the Fisher information matrix $\boldsymbol{F}$. Theorem 1 from [3] highlights that in this Riemannian manifold, $-\boldsymbol{F}^{-1}\nabla\mathcal{L}(\boldsymbol{w})$ is the steepest descent direction of the loss. Thus, if $\langle\tilde{\boldsymbol{g}}, -\boldsymbol{F}^{-1}\nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w})\rangle \leq 0, \tilde{\boldsymbol{g}}$ will also be a valid descent direction with respect to the metric structure of the manifold. The following lemma proves this.

Lemma 4.1. *Let $\boldsymbol{F}^{-1}\boldsymbol{g}$ be the natural gradient of loss function $\mathcal{L}(\boldsymbol{w})$ and $S = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$ be an orthogonal basis. Define $\tilde{\boldsymbol{g}} = \boldsymbol{F}^{-1}\boldsymbol{g} - \sum_i^k \mathrm{proj}_{\boldsymbol{v}_i}(\boldsymbol{F}^{-1}\boldsymbol{g})$, where $\mathrm{proj}_{\boldsymbol{v}_i}(\boldsymbol{F}^{-1}\boldsymbol{g})$ denotes the projection of $\boldsymbol{F}^{-1}\boldsymbol{g}$ onto the $\boldsymbol{v}_i$. Then, $-\tilde{\boldsymbol{g}}$ is also a descent direction for $\mathcal{L}(\boldsymbol{w})$, with respect to the metric structure of the model parameter space.*

The proof for this lemma can be found in Appendix A.

4.5 EKFAC Fisher Approximation

Calculating the exact Fisher information matrix (FIM) and its inverse becomes prohibitively expensive as our model size grows. To make computation tractable, we utilize the Eigenvalue-corrected Kronecker Factorization (EKFAC) [15]. EKFAC works by providing a diagonal approximation to the FIM not in the raw parameter basis, but in the Kronecker-factored eigenbasis, where a diagonal captures most of the curvature information at a very low cost. Each neural net layer's Fisher block is approximated as $\boldsymbol{F} = \boldsymbol{A} \otimes \boldsymbol{B}$ with the Kronecker factors $\boldsymbol{A} = \boldsymbol{Q}_{\boldsymbol{A}}\boldsymbol{\Lambda}_{\boldsymbol{A}}\boldsymbol{Q}_{\boldsymbol{A}}^\top$ and $\boldsymbol{B} = \boldsymbol{Q}_{\boldsymbol{B}}\boldsymbol{\Lambda}_{\boldsymbol{B}}\boldsymbol{Q}_{\boldsymbol{B}}^\top$. We track just the diagonal matrices $\boldsymbol{\Lambda}_{\boldsymbol{A}}$ and $\boldsymbol{\Lambda}_{\boldsymbol{B}}$ in these eigenbases, so each mini-batch update just becomes two small eigenbasis transformations plus a diagonal rescaling. We utilize this approximation of the Fisher matrix in our code implementation. We adapt this EKFAC algorithm [16] in our code implementation for approximating the Fisher matrix, leading to faster and more stable training.

5 Experiments

5.1 Datasets

Our preliminary evaluations include two standard benchmarks utilized extensively in continual learning literature: Permuted MNIST and Rotated MNIST. Permuted MNIST generates K tasks by fixing K independent random permutations of the 784 input pixels. Each task is then the original MNIST images with its pixels reshuffled by one of those permutations, forcing the model to learn to classify under radically different orderings without forgetting prior permutations. Rotated MNIST applies a fixed rotation angle θ_k to every MNIST digit. At task k, all training and test images are rotated by θ_k, which forces the model to continually adapt to new orientations while remembering earlier orientations with different angles.

5.2 Evaluation Criteria

For evaluation purposes, we adapt the same metrics as [12]: average accuracy and average forgetting. Let $a_{T,k}$ denote the accuracy of the model on the $\mathcal{T}_k$ after being trained on the task $\mathcal{T}_T$. Average accuracy (A_T) is the accuracy of the model after its trained on task $\mathcal{T}_T$. It is defined as:

$$A_T = \frac{1}{T}\sum_{k=1}^{T} a_{T,k} \tag{5}$$

Average forgetting (F_T) is the average forgetting the model has after its been trained on task $\mathcal{T}_T$. It is defined as:

$$F_T = \frac{1}{T-1}\sum_{k=1}^{T-1} \max_{t\in\{1,\dots T-1\}} (a_{t,k} - a_{T,k}) \tag{6}$$

5.3 Setup

Details can be found in Appendix B.

5.4 Results

5.4.1 Forgetting and Average Validation Accuracy

Here, we present the average forgetting and average validation accuracy, as described above, of the two baseline methods (OGD and OGD+), as well as our natural gradient extensions to each (ONG and ONG+). Table 1 highlights the average forgetting values and Table 2 highlights the final average validation accuracy. As we can see, simply adding a preconditioner to the OGD algorithm leads to extremely high fogetting and lower accuracy. These results suggest that there is some underlying tension between the Euclidian-style projections of OGD with the Fisher-geometry descent directions. While the reason for this discrepancy is unclear, we are currently investigating this phenomena by using crafted datasets and evaluations.

Table 1: Average forgetting of all the methods we considered (lower is better).

	Permuted MNIST	**Rotated MNIST**
OGD	4.1534	16.1282
OGD+	1.0563	6.0064
ONG	58.7240	28.7360
ONG+	57.5692	27.6238

Table 2: Final validation accuracy, averaged over all tasks, of all the methods we considered.

	Permuted MNIST	**Rotated MNIST**
OGD	82.6604	77.7443
OGD+	85.8746	87.6950
ONG	32.5053	69.0968
ONG+	33.5723	70.1771

5.4.2 Permuted MNIST

Here, we more closely examine our methods' performance on Permuted MNIST. Table 3 lists the final task-wise test accuracies for each method after the model finishes sequentially training on all tasks. The plots in Figure 1 examine the evolution of validation accuracy during training, specifically how per-task accuracies evolve as the model continues to be trained on newer and newer tasks. These finer-grain results show just how quickly the accuracy starts dropping off as we train on newer tasks, and further reinforce the idea that our method of projection is likely misaligned with our metric space of choice.

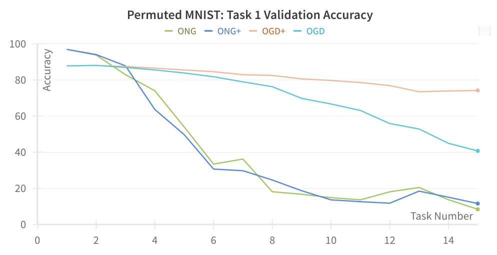
Table 3: Permuted MNIST: Test accuracy of each method on the indicated task after training sequentially on all tasks.

(a) Tasks 1 – 7

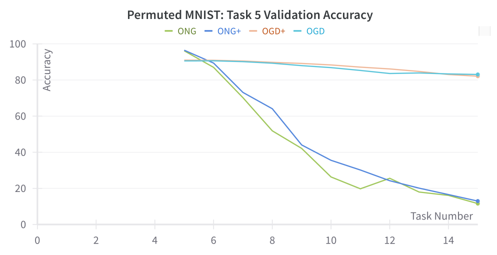
Method	Accuracy						
	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7
OGD	38.91	68.55	71.73	79.71	83.41	87.27	85.83
OGD+	74.69	76.57	76.12	79.42	82.03	84.72	85.16
ONG	5.44	10.22	17.08	14.24	8.17	12.88	15.74
ONG+	13.64	16.48	15.44	16.26	11.00	14.39	19.69

(b) Tasks 8 – 15

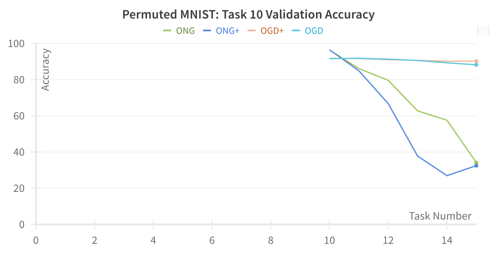
Method	Accuracy							
	Task 8	Task 9	Task 10	Task 11	Task 12	Task 13	Task 14	Task 15
OGD	87.26	88.79	88.59	89.93	91.56	92.02	92.96	93.39
OGD+	87.91	89.03	90.56	90.74	91.92	92.74	93.11	93.39
ONG	23.73	24.75	25.89	44.99	40.09	66.53	81.48	96.35
ONG+	23.86	26.40	33.47	29.67	43.95	54.85	88.15	96.34



(a) Task 1 accuracy throughout tasks



(b) Task 5 accuracy throughout tasks



(c) Task 10 accuracy throughout tasks

Figure 1: Validation Accuracy for tasks 1, 5, and 10 of the Permuted MNIST dataset throughout training. The model is sequentially trained on tasks 1 through 15, and thus the x-axis is presented in terms of number of tasks.

5.4.3 Rotated MNIST

Here, we more closely examine our methods' performance on Rotated MNIST. Table 4 lists the final task-wise test accuracies for each method after the model finishes sequentially training on all tasks. The plots in Figure 2 examine the evolution of validation accuracy during training, specifically how per-task accuracies evolve as the model continues to be trained on newer and newer tasks. While less extreme, we still observe the same sharper dropoff here as well.

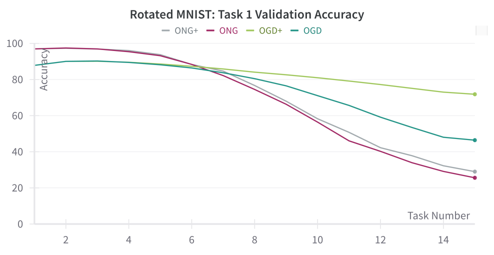
Table 4: Rotated MNIST: Test accuracy of each model on the indicated task after training sequentially on all tasks.

(a) Tasks 1 – 7

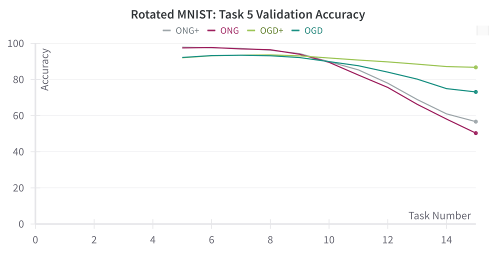
Method	Accuracy						
	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7
OGD	41.83	43.75	53.89	61.19	68.45	74.93	82.01
OGD+	69.67	73.21	79.06	83.27	85.71	86.87	89.59
ONG	24.42	27.60	34.24	41.48	49.81	58.32	67.85
ONG+	27.59	30.30	37.41	43.31	51.86	59.68	68.94

(b) Tasks 8 – 15

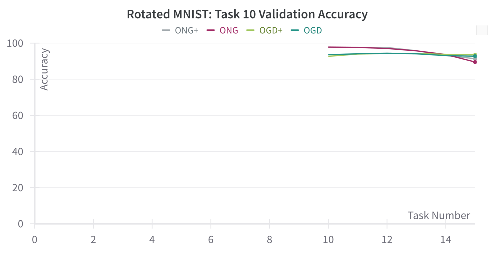
Method	Accuracy							
	Task 8	Task 9	Task 10	Task 11	Task 12	Task 13	Task 14	Task 15
OGD	85.49	89.33	91.97	93.39	94.83	95.03	95.27	94.80
OGD+	90.15	91.91	92.95	93.74	94.83	95.15	94.92	94.40
ONG	75.81	84.03	90.02	93.88	96.41	97.17	97.74	97.68
ONG+	77.10	84.27	89.58	93.64	96.23	97.39	97.58	97.78



(a) Task 1 accuracy throughout tasks



(b) Task 5 accuracy throughout tasks



(c) Task 10 accuracy throughout tasks

Figure 2: Validation Accuracy for tasks 1, 5, and 10 of the Rotated MNIST dataset throughout training. The model is sequentially trained on tasks 1 through 15, and thus the x-axis is presented in terms of number of tasks.

6 Conclusion and Outlook

In this work, we proposed and investigated Orthogonal Natural Gradient Descent (ONG), a novel algorithm that incorporates natural gradients with OGD, a continual learning algorithm.

Our theoretical contributions include proving that ONG maintains descent-direction guarantees under the Fisher metric and detailing an efficient implementation using the EKFAC approximation.

Our empirical evaluation, however, yielded a counterintuitive yet insightful result: ONG underperforms its Euclidean counterpart on standard continual learning benchmarks. Rather than a simple failure of the method, we interpret this as a key finding of our study: **there exists a fundamental tension between Fisher preconditioning and standard orthogonal projections**. This discovery suggests that a naive combination of these methods is not only suboptimal but can be detrimental, indicating that the underlying geometry is not immediately compatible. This finding motivates a clear path for future research focused on reconciling these two perspectives. Our primary hypothesis is that the type of projection we utilize must be chosen to be compatible with the descent directions natural gradient descent uses. We are currently investigating on understanding what the exact failure modes are by crafting synthetic, task-datasets and analyzing the training dynamics. Further future work will involve looking more closely at the evolution of the task-specific gradient subspaces, and the accuracy of the Fisher approximation estimates. Ultimately, we aim to utilize these insights and develop a geometrically-meaningful way of performing "projection" with natural gradients that preserves previously seen gradient directions and enables high accuracy on older tasks.

To that end, incorporating more advanced concepts from Manifold Learning and Information Geometry presents an exciting direction. The concept of parallel transport (see Appendix C for its definition) is particularly promising, as it provides a principled way to move tangent vectors (i.e., gradients) between different points on the parameter manifold [17, 18]. By using parallel transport to bring gradients from previous tasks into the same tangent space as the current gradient, it may be possible to define a geometrically consistent inner product and basis for projection.

Additional avenues for research include a more rigorous study of the parameterization-invariance property, a primary motivator for this work, to understand which kind of model reparameterizations our method is robust to. Finally, we posit that the limitations we observed may be partially attributable to the high task correlation in benchmarks like Permuted and Rotated MNIST. We hypothesize that after developing a more robust, geometrically-aware continual learning method, its benefits will be more pronounced in more challenging and realistic continual learning scenarios involving diverse, uncorrelated task sequences. Future work will extend our empirical validation to these more complex domains.

References

[1] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989. doi: https://doi.org/10.1016/S0079-7421(08)60536-8. URL https://www.sciencedirect.com/science/article/pii/S0079742108605368.

[2] Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks, 2015. URL https://arxiv.org/abs/1312.6211.

[3] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. arXiv:1910.07104, 2019. URL https://arxiv.org/abs/1910.07104.

[4] Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998. doi: 10.1162/089976698300017746. URL https://direct.mit.edu/neco/article/10/2/251/6143/Natural-Gradient-Works-Efficiently-in-Learning.

[5] James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020. URL http://jmlr.org/papers/v21/17-678.html.

[6] Sham M Kakade. A natural policy gradient. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001. URL https://proceedings.neurips.cc/paper_files/paper/2001/file/4b86abe48d358ecf194c56c69108433e-Paper.pdf.

[7] Kaiwen Wu and Jacob R. Gardner. Understanding stochastic natural gradient variational inference, 2024. URL https://arxiv.org/abs/2406.01870.

[8] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):35213526, March 2017. ISSN 1091-6490. doi: 10.1073/pnas.1611835114. URL http://dx.doi.org/10.1073/pnas.1611835114.

[9] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence, 2017. URL https://arxiv.org/abs/1703.04200.

[10] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay, 2017. URL https://arxiv.org/abs/1705.08690.

[11] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks, 2022. URL https://arxiv.org/abs/1606.04671.

[12] Mehdi Abbana Bennani, Thang Doan, and Masashi Sugiyama. Generalisation guarantees for continual learning with orthogonal gradient descent. arXiv:2006.11942, 2020. URL https://arxiv.org/abs/2006.11942.

[13] Mohammad Emtiyaz Khan and Wu Lin. Conjugate-computation variational inference : Converting variational inference in non-conjugate models to inferences in conjugate models, 2017. URL https://arxiv.org/abs/1703.04265.

[14] Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational continual learning. arXiv:1710.10628, 2017. URL https://arxiv.org/abs/1710.10628.

[15] Théo George, Emmanuel MarceauCaron, Brendt Wohlberg, and Mark Sandler. Eigenvaluecorrected kronecker factorization for natural gradient descent. arXiv:1806.03884, 2018. URL https://arxiv.org/abs/1806.03884.

[16] Pattarapo Vantanapath. Natural gradient descent - examples and tutorials. https://github.com/wiseodd/natural-gradients, 2020. Accessed: 2024-05-15.

[17] Max Budninskiy, Gloria Yin, Leman Feng, Yiying Tong, and Mathieu Desbrun. Parallel transport unfolding: A connection-based manifold learning approach. In *Symposium on Geometry Processing*, 2018. URL https://geometry.caltech.edu/pubs/BYFTD18.pdf.

[18] Jiaxiang Li and Shiqian Ma. Federated learning on riemannian manifolds, 2022. URL https://arxiv.org/abs/2206.05668. Manuscript.

A Proof of Lemma 4.1

For $-\tilde{\boldsymbol{g}}$ to be a valid descent direction it should satisfy $\langle -\tilde{\boldsymbol{g}}, \boldsymbol{F}^{-1}\boldsymbol{g}\rangle \leq 0$. We have

$$\langle -\tilde{\boldsymbol{g}},\ \boldsymbol{F}^{-1}\boldsymbol{g}\rangle = \langle -\tilde{\boldsymbol{g}}, \tilde{\boldsymbol{g}} + \sum_{i=1}^{k} \mathrm{proj}_{\boldsymbol{v}_i}(\boldsymbol{F}^{-1}\boldsymbol{g})\rangle \tag{7}$$

$$= -\|\tilde{\boldsymbol{g}}\|^2 - \langle \tilde{\boldsymbol{g}}, \sum_{i=1}^{k} \mathrm{proj}_{\boldsymbol{v}_i}(\boldsymbol{F}^{-1}\boldsymbol{g})\rangle\ . \tag{8}$$

Since $\tilde{\boldsymbol{g}} = \boldsymbol{F}^{-1}\boldsymbol{g} - \sum_{i=1}^{k} \mathrm{proj}_{\boldsymbol{v}_i}(\boldsymbol{F}^{-1}\boldsymbol{g})$ is orthogonal to the space spanned by the vectors in S and each $\mathrm{proj}_{\boldsymbol{v}_i}(\boldsymbol{F}^{-1}\boldsymbol{g})$ lies in this space, $\tilde{\boldsymbol{g}}$ will be orthogonal to $\sum_{i=1}^{k} \mathrm{proj}_{\boldsymbol{v}_i}(\boldsymbol{F}^{-1}\boldsymbol{g})$. Thus, $\langle \tilde{\boldsymbol{g}}, \sum_{i=1}^{k} \mathrm{proj}_{\boldsymbol{v}_i}(\boldsymbol{F}^{-1}\boldsymbol{g})\rangle = 0$. Substituting this into Eq. 8, we arrive at $\langle -\tilde{\boldsymbol{g}},\ \boldsymbol{F}^{-1}\boldsymbol{g}\rangle = -\|\tilde{\boldsymbol{g}}\|^2 \leq 0$. Therefore, $-\tilde{\boldsymbol{g}}$ is a valid descent direction for $\mathcal{L}(\boldsymbol{w})$ while maintaining orthogonality to S. ■

B Setup

We use the official code implementation of [12], which can be obtained here as our starting point for building ONG. We also adapt the EKFAC algorithm provided by [16] for our efficient Fisher information matrix implementation. Both repositories have a MIT license.

We train a randomly-initialized MLP (using Kaiming/He initialization) with depth $L = 3$, widths $n_0 = 784$, $n_1 = n_2 = d = 100$, $n_3 = 10$, and ReLU activation functions. The total number of parameters is $\sim 115,000$.

After some experimentation, we chose 3 epochs per task, a learning rate of 0.001, a memory buffer size of 100 samples per task, and a batch size $B = 32$. No training augmentations were added (though this could be part of a future follow-up). All training and validation was done on a single NVIDIA A100.

The training step logic was modified to keep track of the running estimates needed for the EKFAC algorithm. The optimizer step logic was modified to efficiently use EKFAC for preconditioning the gradient vector.

C Parallel Transport

Definition 1 (Parallel transport)**.** *Given a complete Riemannian manifold* $(\mathcal{M}, g)$ *and two points* $x, y \in \mathcal{M}$*, the* parallel transport

$$P_{x\to y} : T_x\mathcal{M} \longrightarrow T_y\mathcal{M}^1 \tag{9}$$

is the linear isometry satisfying

$$\left\langle P_{x\to y}\,\xi,\ P_{x\to y}\,\zeta\right\rangle_y = \langle \xi, \zeta\rangle_x \quad \forall\, \xi, \zeta \in T_x\mathcal{M}. \tag{10}$$

[1]On a complete Riemannian manifold the geodesic between any two points is unique, so $P_{x\to y}$ is well-defined.