Orthogonal Gradient Descent for Continual Learning

Mehrdad Farajtabar
DeepMind

Navid Azizan[1]
CalTech

Alex Mott
DeepMind

Ang Li
DeepMind

Abstract

Neural networks are achieving state of the art and sometimes super-human performance on learning tasks across a variety of domains. Whenever these problems require learning in a continual or sequential manner, however, neural networks suffer from the problem of *catastrophic forgetting*; they forget how to solve previous tasks after being trained on a new task, despite having the essential capacity to solve both tasks if they were trained on both simultaneously. In this paper, we propose to address this issue from a parameter space perspective and study an approach to restrict the direction of the gradient updates to avoid forgetting previously-learned data. We present the Orthogonal Gradient Descent (OGD) method, which accomplishes this goal by projecting the gradients from new tasks onto a subspace in which the neural network output on previous task does not change and the projected gradient is still in a useful direction for learning the new task. Our approach utilizes the high capacity of a neural network more efficiently and does not require storing the previously learned data that might raise privacy concerns. Experiments on common benchmarks reveal the effectiveness of the proposed OGD method.

1 Introduction

One critical component of intelligence is the ability to learn *continuously*, when new information is constantly available but previously presented information is unavailable to retrieve. Despite their ubiquity in the real world, these problems have posed a long-standing challenge to artificial intelligence (Thrun and Mitchell, 1995; Hassabis et al., 2017).

A typical neural network training procedure over a sequence of different tasks usually results in degraded performance on previously trained tasks if the model could not revisit the data of previous tasks. This phenomenon is called *catastrophic forgetting* (McCloskey and Cohen, 1989; Ratcliff, 1990; French, 1999). Ideally, an intelligent agent should be able to learn consecutive tasks without degrading its performance on those already learned. With the deep learning renaissance (Krizhevsky et al., 2012; Hinton et al., 2006; Simonyan and Zisserman, 2014) this problem has been revived (Srivastava et al., 2013; Goodfellow et al., 2013) with many follow-up studies (Parisi et al., 2019).

One probable reason for this phenomenon is that neural networks are usually trained by Stochastic Gradient Descent (SGD)—or its variants—where the optimizers produce gradients that are oblivious to past knowledge. These optimizers, by design, produce gradients that are purely a function of the current mini-batch (or some smoothed average of a short window of them). This is a desirable feature when the training data is iid, but is not desirable when the training distribution shifts over time. In this paper, we present a system where the gradients produced on a training minibatch can avoid interfering with gradients produced on previous tasks.

The core idea of our approach, Orthogonal Gradient Descent (OGD), is to preserve the previously acquired knowledge by maintaining a space consisting of the gradient directions of the neural network predictions on previous tasks. Any update orthogonal to this gradient space change the output of the network minimally. When training on a new task, OGD projects the loss gradients of new samples perpendicular to this gradient space before applying back-propagation. Empirical results demonstrate that the proposed method efficiently utilizes the high capacity of the (often over-parameterized) neural network to learn the new data while minimizing the interference with the previously acquired knowledge. Experiments on three common continual learning benchmarks substantiate that OGD achieves state-of-the-art performance without the need to store the historical data.

Correspondence to farajtabar@google.com.
[1] Work done during an internship at DeepMind.

2 Preliminaries

Consider a *continual learning* setting in which tasks $\{T_1, T_2, T_3, \ldots\}$ arrive sequentially. When a model is being trained on task T_k, any data from previous tasks $\{T_t \mid t < k\}$ is inaccessible. Each data point $(x, y) \in T_k$ is a pair consists of input $x \in \mathbb{R}^d$ and a label y. For a c-class classification, y is a c-dimensional one hot vector. The prediction of the model on input x is denoted by $f(x; w)$, where $w \in \mathbb{R}^p$ are parameters (weights) of the model (neural network). For classification problems, $f(x; w) \in \mathbb{R}^c$ where $f_j(x; w)$ is the j-th logit associated to j-th class.

The total loss on the training set (empirical risk) for task t is denoted by

$$L_t(w) = \sum_{(x,y)\in T_t} L_{(x,y)}(w), \tag{1}$$

where the per-example loss is defined as

$$L_{(x,y)}(w) = \ell(y, f(x; w)), \tag{2}$$

and $\ell(\cdot, \cdot)$ is a differentiable non-negative loss function. For classification problems, a softmax cross entropy loss is commonly used, *i.e.*,

$$\ell(y, f(x; w)) = -\sum_{j=1}^{c} y_j \log a_j, \tag{3}$$

where $a_j = \exp f_j(x; w) / \sum_k \exp f_k(x; w)$ is the j-th softmax output.

Two objects that frequently appear throughout the development of our method are the gradient of the *loss*, $\nabla L_{(x,y)}(w) \in \mathbb{R}^p$, and the gradient of the *model*, $\nabla f(x; w) \in \mathbb{R}^{p\times c}$, which are both with respect to w, and it is critically important to distinguish the two. In fact, the gradient of the loss, using the chain rule, can be expressed as

$$\nabla L_{(x,y)}(w) = \nabla f(x; w) \ell'(y, f(x; w)), \tag{4}$$

where $\ell'(\cdot, \cdot) \in \mathbb{R}^c$ denotes the derivative of $\ell(\cdot, \cdot)$ with respect to its second argument, and $\nabla f(x; w)$ is the gradient of the model f with respect to its second argument (*i.e.*, the parameters). For the classification problem with cross entropy softmax loss, we have

$$\nabla f(x; w) = [\nabla f_1(x; w); \ldots; \nabla f_c(x; w),], \tag{5}$$

and the derivative of the loss becomes

$$\ell'(y, f(x; w)) = [a_1 - y_1, \ldots, a_c - y_c]^\top, \tag{6}$$

where, $\nabla f_j(x; w) \in \mathbb{R}^c$ is the gradient of the j-th logit with respect to parameters.

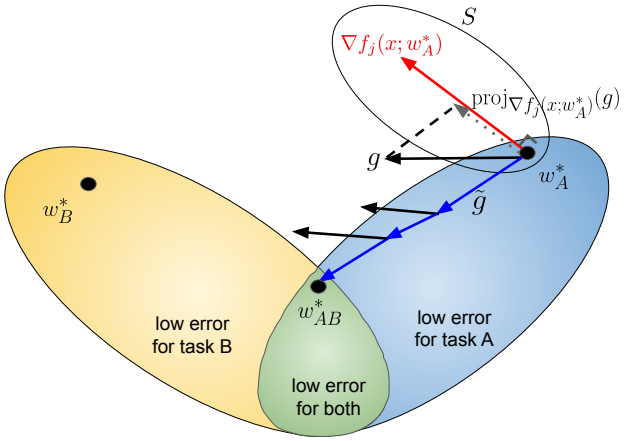

Figure 1: An illustration of how Orthogonal Gradient Descent corrects the directions of the gradients. g is the original gradient computed for task B and $\tilde{g}$ is the projection of g onto the orthogonal space *w.r.t* the gradient $\nabla f_j(x; w_A^*)$ computed at task A. Moving within this (blue) space allows the model parameters to get closer to the low error (green) region for both tasks.

3 Orthogonal Gradient Descent

Catastrophic forgetting happens in neural networks when the gradient updates with respect to a new task are applied to the model without considering previous tasks. We propose the Orthogonal Gradient Descent method for mitigating this problem, which is based on modifying the direction of the updates to account for *important directions* of previous tasks. Figure 1 shows an illustration of the core idea of OGD which constrains the parameters to move within the orthogonal space to the gradients of previous tasks.

Suppose that the model has been trained on the task A in the usual way until convergence to a parameter vector w_A^* so that the training loss/error is small or zero, and consider the effect of a small update to W_A^*. In the high-dimensional parameter space of the model, there could be update directions causing large changes in the predictions from $x \in T_A$, while there also exist updates that minimally affect such predictions. In particular, moving locally along the direction of $\pm\nabla f_j(x; w)$ leads to the biggest change in model prediction $f_j(x; w)$ given any sample x, while moving orthogonal to $\nabla f_j(x; w)$ leads to the least change (or no change, locally) to the prediction of x. Supposing task A has n_A data points in the stochastic gradient descent setting, there will be $n_A \times c$ gradient directions $\nabla f_j(x; w)$. In order to guarantee the least change to the predictions on task A, for an update towards task B, the update has to be orthogonal to the $n_A \times c$ directions $\{\nabla f_j(x; w)\}_{x\in T_A, j=1\ldots c}$.

Denoting the gradient of the loss for task B (which can be either stochastic, batch, or full) by g, we propose to "orthogonalize" it in a way that the new direction $\tilde{g}$ satisfies the above requirement, i.e.,

$$\tilde{g} \perp \nabla f_j(x; w), \quad \forall x \in T_A, j = 1 \dots c \; . \tag{7}$$

In this case, moving along direction $\tilde{g}$ makes the least change to the neural network predictions on the previous task. As a result, we utilize the high capacity of neural networks more effectively. We know that a neural network is part of a high dimensional parameter space (larger than or comparable to the number of data points), so there always exist a direction that conforms to the orthogonality condition.

In continual leaning, while processing T_B, one does not have access to T_A anymore to compute $\nabla f_j(x; w)$ at the current parameter w. This means that, as an inherent limitation, we are unable to compute the exact directions that will produce least changes on Task A during training on Task B. To tackle this issue note that the neural networks are often overparameterized, which implies that in a close vicinity of the optimum parameter for task A, there lies optimum parameters for both tasks A and B (Azizan and Hassibi, 2018; Li and Liang, 2018; Allen-Zhu et al., 2018; Azizan et al., 2019). For any parameter w in that neighborhood, we can basically approximate $\nabla f(x; w) \approx \nabla f(x; w^*_A)$ for all $x \in T_A$. Therefore, we can use $\nabla f(x; w^*_A)$ as proxy and satisfy

$$\tilde{g} \perp \nabla f_j(x; w^*_A), \quad \forall x \in T_A, j = 1 \dots c, \tag{8}$$

for all (batch) loss gradients $\tilde{g}$ of task B. One can compute and store $\nabla f(x; w^*_A)$ for all $x \in T_A$ when training on task A is done and task B is introduced.

In practice, one does not need all $n_A \times c$ directions $\{\nabla f_j(x; w)\}_{x \in T_A, j=1 \dots c}$ to preserve the previously learned information. For example, per sample x, we can compute the gradient with respect to the average of the logits rather than use the individual logits themselves. We call this OGD-AVE in contrast to OGD-ALL. Another alternative is to select the single logit corresponding to the ground truth label. For data point x coming from k-th class ($y_k = 1$), we try to only keep $\nabla f_k(x; w)$ invariant. This alternative referred to as OGD-GTL. Both OGD-GTL and OGD-AVE reduce the storage size by a factor of c. We use OGD-GTL in all of our following experiments and also empirically observe that OGD-GTL slightly outperforms the other variants of OGD (*c.f* appendix A.1). To further control the amount of memory required for this process, we store only a subset of gradients from each task in our experiments (200 for the Mnist experiments). While this potentially misses some information, we find in practice that it is sufficient and that increasing the collection size beyond this provides diminishing returns. One downside of this method (similar to other state-of-the-art methods such as (Chaudhry et al., 2018; Lopez-Paz and Ranzato, 2017; Riemer et al., 2018)) is that the required storage size grows with the number of tasks. An interesting extension to our method is to dynamically remove less significant directions from set S or perform principal component analysis on the gradient space, which are left for future work.

Algorithm 1 Orthogonal Gradients Descent

Input Task sequence $T_1, T_2, T_3, \dots$ learning rate η
Output The optimal parameter w.

Initialize $S \leftarrow \{\}$; $w \leftarrow w_0$
for Task ID $k = 1, 2, 3, \dots$ **do**
  repeat
    $g \leftarrow$ Stochastic/Batch Gradient for T_k at w
    $\tilde{g} = g - \sum_{v \in S} \text{proj}_v(g)$
    $w \leftarrow w - \eta \tilde{g}$
  until convergence
  for $(x, y) \in T_t$ and $k \in [1, c]$ s.t. $y_k = 1$ **do**
    $u \leftarrow \nabla f_k(x; w) - \sum_{v \in S} \text{proj}_v(\nabla f_k(x; w))$
    $S \leftarrow S \cup \{u\}$

We now proceed to formally introduce OGD-GTL. Task B's loss gradients should be perpendicular to the *space* of all previous model gradients, namely

$$S = \text{span}\{\nabla f_k(x, w^*_A) \mid (x, y) \in T_A \wedge k \in [1, c] \wedge y_k = 1\}.$$

We compute the orthogonal basis for S as $\{v_1, v_2, \dots\}$ using the Gram-Schmidt procedure on all gradients *w.r.t.* samples $(x_i, y_i) \in T_A$ in task A. We iteratively project them to the previously orthogonalized vectors:

$$v_1 = \nabla f_{k_1}(x_1; w^*_A) \; ,$$
$$v_i = \nabla f_{k_i}(x_i; w^*_A) - \sum_{j<i} \text{proj}_{v_j}(\nabla f(x_i; w^*_A)),$$

where, k_i represents the ground-truth index such that $y_{i,k_i} = 1$, and $\text{proj}_v(u) = \frac{\langle u,v \rangle}{\langle v,v \rangle} v$ is the projection (vector) of u in the direction of v. Given the orthogonal basis $S = \{v_1, \dots, v_{n_A}\}$ for the gradient subspace of task A, we modify the original gradients g of task B to new gradients $\tilde{g}$ orthogonal to S, *i.e.*,

$$\tilde{g} = g - \sum_{i=1}^{n_A} \text{proj}_{v_i}(g) \; . \tag{9}$$

The new direction $-\tilde{g}$ is still a descent direction (*i.e.* $\langle -\tilde{g}, g \rangle \leq 0$) for task B meaning that $\exists \; \epsilon > 0$ such that for any learning rate $0 < \eta < \epsilon$, taking the step $\eta \tilde{g}$ reduces the loss.

Lemma 3.1. *Let g be the gradient of loss function $L(w)$ and $S = \{v_1, \dots, v_n\}$ is the orthogonal basis.*

Let $\tilde{g} = g - \sum_i^k \text{proj}_{v_i}(g)$. *Then,* $-\tilde{g}$ *is also a descent direction for* $L(w)$.

Proof. For a vector u to be a descent direction it should satisfy $\langle u, g\rangle \leq 0$. To begin with, we have

$$\langle -\tilde{g},\ g\rangle = \langle -\tilde{g}, \tilde{g} + \sum_{i=1}^{k} \text{proj}_{v_i}(g)\rangle \tag{10}$$

$$= -\|\tilde{g}\|^2 - \langle \tilde{g}, \sum_{i=1}^{k} \text{proj}_{v_i}(g)\rangle\ . \tag{11}$$

Since $\tilde{g} = g - \sum_{i=1}^{k} \text{proj}_{v_i}(g)$ is orthogonal to the space spanned by S and $\sum_{i=1}^{k} \text{proj}_{v_i}(g)$ is a vector spanned by S, hence $\langle \tilde{g}, \sum_{i=1}^{k} \text{proj}_{v_i}(g)\rangle = 0$. Substituting this into Eq. 11, we have $\langle -\tilde{g},\ g\rangle = -\|\tilde{g}\|^2 \leq 0$. Therefore, $-\tilde{g}$ is a descent direction for $L(w)$ while being perpendicular to S. □

We can easily extend the method to handle multiple tasks. Algorithm 1 presents this general case. In this work, we apply the proposed Orthogonal Gradient Descent (OGD) algorithm to continual learning on consecutive tasks. Its application potentially goes beyond this special case and can be utilized whenever one wants the gradient steps minimally interfere with the previous learned data points and potentially reduce the access or iterations over them.

It is worth reiterating the distinction made in Section 2 between using the gradient of the logits—as OGD does—and using the gradient of the loss—as many other methods do, including the A-GEM baseline (Chaudhry et al. (2018)) in the next section. As Equation (4) indicates, the gradient of the loss $\nabla L_{(x,y)}(w)$ can be zero or close to zero for the examples that are well fitted ($\ell'(y, f(x; w)) \approx 0$) carrying effectively low information on the previous tasks. In contrast, OGD works directly with the model (through its gradeint $\nabla f(x; w)$) which is the essential information to be preserved.

4 Experiments

We performed experiments in this section on three continual learning benchmark: *Permuted Mnist*, *Rotated Mnist*, and *Split Mnist*.

Baselines. We considered the following baselines for comparison purposes. (1) *EWC* (Kirkpatrick et al., 2017): one of the pioneering regularization based methods that uses fisher information diagonals as important weights. (2) *A-GEM* (Chaudhry et al., 2018): using loss gradients of stored previous data in an inequality constrained optimization. (3) *SGD*: Stochastic Gradient Descent optimizing tasks one after the other. It can be seen as lower bound telling us what happens if we do nothing to explicitly retain information from the previous task(s). (4) *MTL*: Multi Task Learning baseline using stochastic gradient descent with full access to previous data. In this setting, during task T_t, we trained the model on batches containing all data $T_{\leq t}$. This can be considered a sort of upper bound on the performance.

Setup. We used a consistent training setup for all Mnist experiments so that we can directly compare the effects of the model across tasks and methods. We always trained each task for 5 epochs. The number of epochs was chosen to achieve saturated performance on the first task classification problem. The performance numbers do not change substantially when trained for more epochs and, crucially, the relative performance between the different methods is identical with more training epochs. We used a batch size of 10 similar to Chaudhry et al. (2018); Lopez-Paz and Ranzato (2017). We found that batch size was not a strong factor in the performance, other than in its interplay with the number of epochs. For fewer than 5 epochs, the batch size had a noticeable effect because it significantly changed the number of batch updates.

Large learning rates do degrade the performance of OGD (the larger the learning rate, the more likely a gradient update violates the locality assumption). We chose a learning rate of 10^{-3}, consistent with other studies (Kirkpatrick et al., 2017; Chaudhry et al., 2018), and small enough that decreasing it further did not improve the performance. For all experiments the same architecture is used. The network is a three-layer MLP with 100 hidden units in two layers and 10 logit outputs. Every layer except the final one uses ReLU activation. The loss is Softmax cross-entropy, and the optimizer is stochastic gradient descent. This setting is similar to previous works (Chaudhry et al., 2018; Kirkpatrick et al., 2017). At the end of every task boundary we performed some processing required by the method. For OGD, this means computing the orthogonal gradient directions as described in Section 3. For A-GEM, this means storing some examples from the ending task to memory. For EWC, this means freezing the model weights and computing the fisher information. Both OGD and A-GEM need an storage. A-GEM for actual data points and OGD for the gradients of the model on previous tasks. We set the storage size for both methods to 200. Last but not least, in all the experiments the mean and standard deviation of the *test error* on the hold out Mnist test set are demonstrated using 10 independent random runs for 2 and 3 task experiments and 5 independent runs for 5 task experiments.

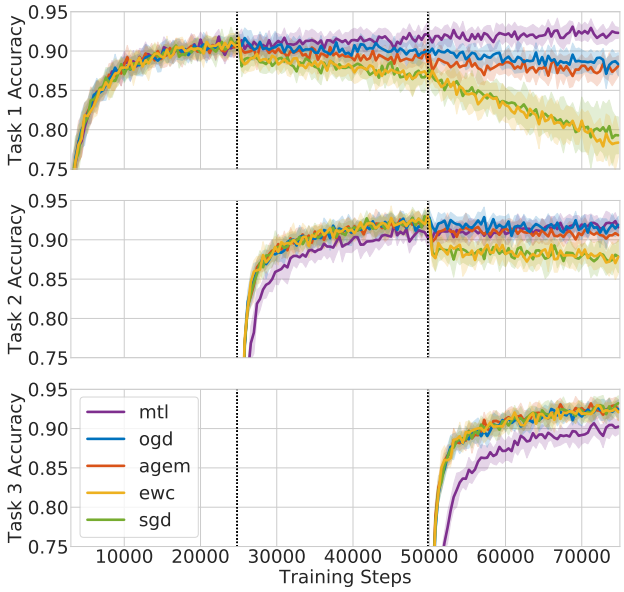

Figure 2: Performance of different methods on permuted Mnist task. 3 different permutations (p_1, p_2, and p_3) are used and the model is trained to classify Mnist digits under permutation p_1 for 5 epochs, then under p_2 for 5 epochs and then under p_3 for 5 epochs. The vertical dashed lines represent the points in the training where the permutations switch. The top plot reports the accuracy of the model on batches of the Mnist test set under p_1; the middle plot, under p_2; and the bottom plot under p_3. The y-axis is truncated to show the details. Note that MTL represents a setting where the model is directly trained on all previous tasks. Because we keep constant batch size and number of epochs, the MTL method effectively sees one third of the task 3 data that other methods do. This is the reason that MTL learns slower on task 3 than other methods.

4.1 Permuted Mnist

We tested our method on the Permuted Mnist setup described in (Goodfellow et al., 2013) and utilized in (Kirkpatrick et al., 2017; Chaudhry et al., 2018) too. In this setup we generated a series of 3 permutations p_1, p_2, and p_3 that shuffle the pixels in an Mnist image. We designated task T_i as the problem of classifying Mnist digits that have been shuffled under permutation p_i. We chose these permutations randomly so each task is equally hard and so the difference in accuracy between examples from task 1 and examples from task 3 after task 3 has been trained is a measure of how much the network is able to remember p_1.

Figure 2 shows the accuracy of OGD and baselines on the Permuted Mnist task. The plot shows that OGD retains performance on task 1 examples as well as A-GEM even after training on task 3. Both methods perform slightly worse than a model that is able to train on all previous tasks (MTL), but significantly better than the naive sequential model (SGD) and than EWC.

	Accuracy ± Std. (%)				
	Task 1	**Task 2**	**Task 3**	**Task 4**	**Task 5**
MTL	93.2 ± 1.3	91.5 ± 0.5	91.3 ± 0.7	91.3 ± 0.6	88.4 ± 0.8
OGD	79.5 ± 2.3	$\mathbf{88.9} \pm 0.7$	$\mathbf{89.6} \pm 0.3$	$\mathbf{91.8} \pm 0.9$	92.4 ± 1.1
A-GEM	$\mathbf{85.5} \pm 1.7$	87.0 ± 1.5	$\mathbf{89.6} \pm 1.1$	91.2 ± 0.8	$\mathbf{93.9} \pm 1.0$
EWC	64.5 ± 2.9	77.1 ± 2.3	80.4 ± 2.1	87.9 ± 1.3	93.0 ± 0.5
SGD	60.6 ± 4.3	77.6 ± 1.4	79.9 ± 2.1	87.7 ± 2.9	92.4 ± 1.1

Table 1: *Permuted Mnist*: The accuracy of models for test examples from the indicated class after being trained on all tasks in sequence, except the multi-task setup (MTL). The best continual learning results are highlighted in **bold**.

We extended this experiments to 5 permutations and tasks in the same manner. For this experiment, we evaluated the classifier after training had completed (at the end of task 5) and measured the accuracy for examples from each of task $1 \ldots 5$. Table 4.1 reports these accuracies for OGD and the baseline training methods. The results suggest that the overall performance of OGD is significantly better than EWC and SGD while being on par with A-GEM.

4.2 Rotated Mnist

We further evaluated our approach on identifying rotated Mnist digits. The training setup is similar to Permuted Mnist except that instead of arbitrary permutation, we used fixed rotations of the Mnist digits. Here we started with a two task problem: task 1 is to classify standard Mnist digits and then task 2 is to classify those digits rotated by a fixed angle.

Figure 3 shows the accuracy of the model when classifying task 1 examples (normal, un-rotated Mnist digits) after the end of training on task 2 (rotated Mnist digits). We report this as a function of the angle of rotation. One can see that, as the angle of rotation increases, the task becomes harder. Even in this harder setting, we still observe that OGD and A-GEM exhibit similar levels of performance.

In the same way as the previous experiment, we extended the rotated Mnist experiment to more tasks by training a classifier on 5 rotated Mnist tasks with increasing angle of rotation. We defined the tasks as classification under angles of $T_1 = Rot(0°)$, $T_2 = Rot(10°)$, $\ldots$, $T_5 = Rot(40°)$, and train the models in that order. Table 2 shows the accuracy of the fully-trained model at classifying examples from each tasks. We can observe that OGD outperforms other methods on 10, 20, and 30 degree rotations.

4.3 Split Mnist

We also tested OGD in a setting where the labels between task 1 and task 2 are disjoint. We followed the

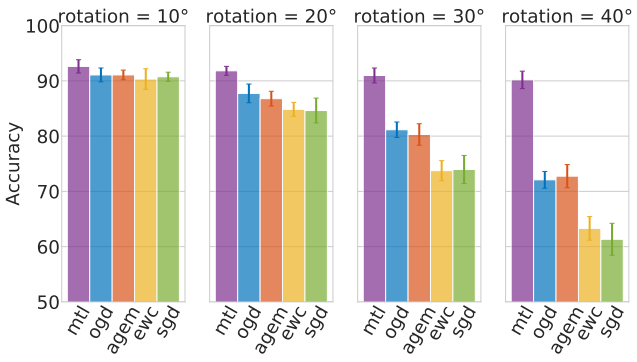

Figure 3: *Rotated Mnist*: Accuracies of multiple continual learning methods. Every classifier is trained for 5 epochs on standard Mnist and then trained for another 5 epochs on a variant of Mnist whose images are rotated by the specified angle. The accuracy is computed over the entire original (un-rotated) Mnist test set after the model being trained on the rotated dataset. Each bar represents the mean accuracy over 10 independent runs and the error bars reflect their standard deviations. MTL represents the (non-continual) multi-task learning setting where the model is trained with the combined data from all previous tasks.

	Accuracy ± Std. (%)				
	Task 1	**Task 2**	**Task 3**	**Task 4**	**Task 5**
MTL	92.1 ± 0.9	94.3 ± 0.9	95.2 ± 0.9	93.4 ± 1.1	90.5 ± 1.5
OGD	$\mathbf{75.6} \pm 2.1$	$\mathbf{86.6} \pm 1.3$	$\mathbf{91.7} \pm 1.1$	94.3 ± 0.8	93.4 ± 1.1
A-GEM	72.6 ± 1.8	84.4 ± 1.6	91.0 ± 1.1	93.9 ± 0.6	$\mathbf{94.6} \pm 1.1$
EWC	61.9 ± 2.0	78.1 ± 1.8	89.0 ± 1.6	94.4 ± 0.7	93.9 ± 0.6
SGD	62.9 ± 1.0	76.5 ± 1.5	88.6 ± 1.4	$\mathbf{95.1} \pm 0.5$	94.1 ± 1.1

Table 2: *Rotated Mnist*: The accuracy of models for test examples from the indicated class after being trained on all tasks in sequence, except the multi-task setup (MTL). The best continual learning results are highlighted in **bold**.

setup for split Mnist laid out in Zenke et al. (2017) with some variations. We defined a set of tasks $T_1 \ldots T_N$, with task T_i defined by a series of integers $t_i^1 \ldots t_i^{k_i}$ with $0 \leq t_i^j \leq 10$ and $t_i^j = t_{i'}^{j'}$ if and only if $i = i'$ and $j = j'$. For each task T_i, then, the task is to classify Mnist digits with labels in $\{t_i^j\}$.

Because a given task does not contain all labels, we used a slightly different architecture for this task compared to other tasks. Instead of having a single output layer containing 10-logits for all the Mnist classes, we used separate heads for each task, where each head has the same number of logits as there are classes in the associated task. This means that, for each task T_i, the softmax and cross-entropy calculation only runs over the logits and labels $\{t_i^j\}$. We found that this model has higher performance under all methods than a model using a joint head.

We began with a two task classification problem, where the Mnist dataset is split into two disjoint sets

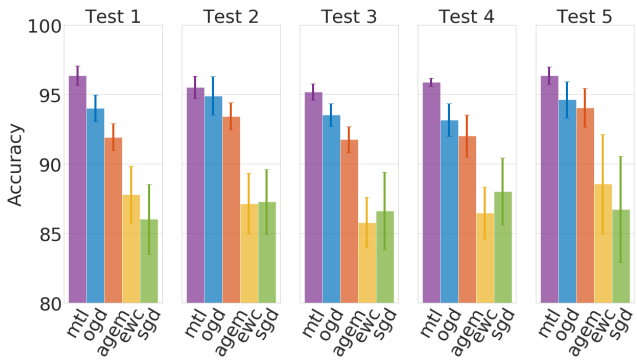

Figure 4: *Split Mnist*: Accuracies of multiple continual learning methods. The training regime is the same as that of Figure 2. The reported value is the accuracy on task 1 after the model being trained on task 2. Different plots correspond to different configurations, *i.e.*, different partitions of the Mnist labels into task 1 and task 2.

	Accuracy ± Std. (%)				
	Task 1	**Task 2**	**Task 3**	**Task 4**	**Task 5**
MTL	99.6 ± 0.2	99.8 ± 0.1	98.8 ± 0.2	98.2 ± 0.4	99.1 ± 0.2
OGD	$\mathbf{98.6} \pm 0.8$	$\mathbf{99.5} \pm 0.1$	$\mathbf{98.0} \pm 0.5$	$\mathbf{98.8} \pm 0.5$	99.2 ± 0.3
A-GEM	92.9 ± 2.6	96.3 ± 2.1	86.5 ± 1.6	92.3 ± 2.3	99.3 ± 0.2
EWC	90.2 ± 5.7	98.9 ± 0.2	91.1 ± 3.5	94.4 ± 2.0	99.3 ± 0.2
SGD	88.2 ± 5.9	98.4 ± 0.9	90.3 ± 4.5	95.2 ± 1.0	$\mathbf{99.4} \pm 0.2$

Table 3: *Split Mnist*[2]: The accuracy of models for test examples from the indicated class after being trained on all tasks in sequence, except the multi-task setup (MTL). The best continual learning results are highlighted in **bold**.

each containing 5 labels. The tasks are then just to classify examples from the set associated with each task. Figure 4 shows the accuracy of the fully-trained model to classify images from task T_1. We report the results for 5 different partitions of the labels into the task sets, to ensure that the partition does not have a strong effect on the results. In all cases, we observe the OGD performs the best, beating A-GEM again. We also observe that the performance order is preserved across different configurations of the experiment.

We again generalized this experiment to a longer sequence of tasks by splitting Mnist into 5 tasks, each with two classes. We used a multi-headed architecture as in the 2 task case. We report the accuracy of the fully trained model on examples from each of the 5 classes in Table 3. As in the previous case, we evaluated this on multiple partitions of the labels; the results from other partitions are shown in the appendix. In this setting, OGD performs very closely to the multi-task training benchmark and consistently outperforms the other baselines.

[2]The accuracy for different assignments of labels to tasks in Table 3 can be found in Appendix A.3.

5 Related Work

There is a growing interest in measuring catastrophic forgetting (Toneva et al., 2018; Kemker et al., 2018), evaluating continual learning algorithms (Farquhar and Gal, 2018; Hayes et al., 2018; Díaz-Rodríguez et al., 2018; De Lange et al., 2019; Hsu et al., 2018), and understating this phenomenon (Nguyen et al., 2019; Farquhar and Gal, 2019). The existing work on alleviating catastrophic forgetting can be divided into a few categories.

The *expansion* based methods allocate new neurons or layers or modules to accommodate new tasks while utilizing the shared representation learned from previous ones. Rusu et al. (2016) proposed progressive neural networks in which parameters for the original task are untouched while the architecture is expanded by allocating new sub-networks with fixed capacity to be trained on the new task. Similarly, Xiao et al. (2014) proposed a method in which the network not only grows in capacity, but forms a hierarchical structure as new tasks arrive at the model. Yoon et al. (2018) proposed a dynamically expanding network that either retrain or expand the network capacity upon arrival of a new task with only the necessary number of units by splitting/duplicating units and timestamping them. Draelos et al. (2017) used auto-encoder to dynamically decide to add neurons for samples with high loss and whether the older data needs to be retrained or not. Along this idea Jerfel et al. (2019) proposed to use Dirichlet process mixture of hierarchical Bayesian models over the parameters of neural networks to dynamically cope with the new tasks. Recently, Li et al. (2019b) proposed to utilize the neural architecture search to find the optimal structure for each of the sequential tasks. These methods avoid storing data and are aligned with neurogenesis in the brain Aimone et al. (2009) but may be complex for the current neural network libraries.

In the *regularization* based approaches, catastrophic forgetting is tackled by imposing constraints on the weight updates of the neural network according to some importance measure for previous tasks. The difference lies in the way how importance weights are computed. In Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017) the importance weights are the diagonal values of the Fisher information matrix which approximates the posterior distribution of the weights. Along this Bayesian perspective, Titsias et al. (2019) proposed to work over the function space rather than the parameters of a deep neural network to avoid forgetting a previous task by constructing and memorizing an approximate posterior belief over the underlying task-specific function. Employing other Bayesian techniques, Nguyen et al. (2017) combine online variational inference and recent advances in Monte Carlo methods. Ritter et al. (2018) recursively approximate the posterior after every task with a Gaussian Laplace approximation of the Hessian for continual learning, and Ebrahimi et al. (2019) used uncertainty measures to help continual learning. Schwarz et al. (2018) proposed a cycle of active learning (progression) followed by consolidation (compression) that requires no architecture growth and no access to or storing of previous data or tasks which is similar in spirit to distillation based methods of continual learning Li and Hoiem (2017); Hu et al. (2018). Knowledge Distillation (Hinton et al., 2015) and its many variants(Romero et al., 2014; Mirzadeh et al., 2019) are useful to retain the previous information.

In (Zenke et al., 2017) each parameter accumulates task relevant information over time, and exploits this information to rapidly store new tasks without forgetting old ones. Lee et al. (2017) incrementally match the moment of the posterior distribution of the neural network trained on the first and the second task to regularize its update on the latter. Other works along this line are (Aljundi et al., 2018; Kolouri et al., 2019) which penalize the weights based on a Hebbian like update rule. These approaches are well motivated by neuro-biological models of memory Fusi et al. (2005); Kaplanis et al. (2018) and are computationally fast and do not require storing data. However, these *consolidated* weights reduce the degree of freedom of the neural network. In other words, they decrease the effective volume of parameter space to search for a configuration that can satisfy both the old and new tasks.

The *repetition* based methods employ memory systems that store previous data or, alternatively, train a generative model for the first task and replay them interleaved with samples drawn from the new task. Shin et al. (2017); Kamra et al. (2017); Zhang et al. (2019); Rios and Itti (2018) learned a generative model to capture the data distribution of previous tasks, along with the current task's data to train the new model so that the forgetting can be alleviated. Lüders et al. (2016) used a Neural Turing Machine that enables agents to store long-term memories by progressively employing additional memory components. In the context of Reinforcement learning Rolnick et al. (2018) utilized on-policy learning on fresh experiences to adapt rapidly to new tasks, while using off-policy learning with behavioral cloning on replay experience to maintain and modestly enhance performance on past tasks. Lopez-Paz and Ranzato (2017) proposed Gradient Episodic Memory (GEM) to efficiently use an episodic storage by following loss gradients on incoming task to the maximum extent while altering them so that they do not interfere with past memories. While minimizing

the loss on the current task GEM treats the losses on the episodic memories of previous tasks as inequality constraints, avoiding their increase but allowing their decrease. Chaudhry et al. (2018) improved GEM by changing the loss function and proposed dubbed Averaged GEM (A-GEM), which enjoys the same or even better performance. Riemer et al. (2018) combined experience replay with optimization based meta-learning to enforce gradient alignment across examples in order to learn parameters that make interference based on future gradients less likely. A few other works have utilized gradient information to protect previous knowledge. He and Jaeger (2018) proposed a variant of the back-propagation algorithm named conceptor-aided backprop that shields gradients against degradation of previously learned tasks. Zeng et al. (2018) ensure that gradient updates occur only in the orthogonal directions to the input of previous tasks. This class of methods also have their root in neuroscience (Kumaran and McClelland, 2012) making training samples as identically distributed as possible. However, they need to store a portion of the data or learning a generative model upon them, which might not be possible in some settings, *e.g.*, with user data when privacy matters. Moreover, many of these methods work with the gradients of the loss, which can be close to zero for many samples and therefore convey less information on previous tasks. In contrast, we work with the gradients of the model (logits or predictions) which is the actual knowledge to be preserved on the course of continual learning. By providing more effective shield of gradients through projecting to the space of previous model gradients, we achieve better protection to previously acquired knowledge, yielding highly competitive results in empirical tests compared to others.

Continual Learning as a sub-field in AI has close connection and ties to other recent efforts in machine learning. *Meta learning* algorithms use a data-driven inductive bias to enhance learning new tasks (Jerfel et al., 2019; He and Jaeger, 2018; Vuorio et al., 2018; Al-Shedivat et al., 2017; Riemer et al., 2018). *Few shot learning* also serves the same purpose and can be leveraged in continual learning (Wen et al., 2018; Gidaris and Komodakis, 2018) and vice versa. The way we treat previous knowledge (i.e. through the model prediction gradients not the actual data) is also related *differential private learning* (Wu et al., 2017; Li et al., 2018; Pihur et al., 2018; Han et al., 2018) and *federated learning* (Bonawitz et al., 2019; Smith et al., 2017; Vepakomma et al., 2018). *Multi-task learning* (Sorokin and Burtsev, 2019), *curriculum learning* (Bengio et al., 2009), and *transfer learning* (Pan and Yang, 2009; Li et al., 2019a) are other related areas helpful to develop better continual learning machines that do not catastrophically forget previous experiences.

6 Conclusion and Outlook

In this paper, we propose to project the current gradient steps to the orthogonal space of neural network predictions on previous data points. The goal is to minimally interfere with the already learned knowledge while gradually stepping towards learning new tasks. We have demonstrated that our method matches or exceeds other state-of-the-art methods on a variety of benchmark experiments. We have observed that OGD is able to retain information over many tasks and achieved particularly strong results on the split Mnist benchmark.

There are several avenues for future study based on this technique. Firstly, because we cannot store gradients for the full datasets there is some forgetting happening. Finding a way to store more gradients or prioritize the important directions would improve the performance. One can also maintain higher-order derivatives of the model for a more accurate representation of previously learned knowledge, at the expense of more memory and computation. Secondly, we have observed that all methods (including ours) fail considerably when the tasks are dissimilar (for example rotations larger than 90 degrees for the Mnist task). This calls for a lot more future research to be invested in this important yet under-explored problem of continual learning. Thirdly, it is observed that our method is sensitive to the learning rate and it sometimes fail to produce comparable results to A-GEM for large learning rates. It's aligned with our expectation that the learning rate is determining the locality and the neighborhood of the search. The gradients of the model predictions at optimal points are a good approximation for the gradients on others if they lie in a close neighborhood. Further work on coping with this would allows OGD to apply to settings where higher learning rates are desired. Another interesting direction for future research is to extend this idea to other types of optimizers such as Adam or Adagrad.

Finally, it is worth noting that the implications of the proposed Orthogonal Gradient Descent goes beyond the standard continual learning setup we described. Firstly, it does not require tasks to be identified and distinguished. OGD minimally interfere with previously seen data points no matter what class they belong to. This makes it applicable when the task shift does not arrive as a distinct event, but rather a gradual shift (He et al., 2019). Moreover, it is applicable to standard learning paradigm where one does not have the luxury of iterating over numerous epochs, as it can preserve information that has not yet been strongly encoded in the weights of the network. A principled extension and verification on common gradient neural network training methods is left as future work.

Acknowledgements

The authors would like to thank Dilan Gorur, Jonathan Schwarz, Jiachen Yang, and Yee Whye Teh for the comments and discussions.

References

Aimone, J. B., Wiles, J., and Gage, F. H. (2009). Computational influence of adult neurogenesis on memory encoding. *Neuron*, 61(2):187–202.

Al-Shedivat, M., Bansal, T., Burda, Y., Sutskever, I., Mordatch, I., and Abbeel, P. (2017). Continuous adaptation via meta-learning in nonstationary and competitive environments. *arXiv preprint arXiv:1710.03641*.

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., and Tuytelaars, T. (2018). Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154.

Allen-Zhu, Z., Li, Y., and Song, Z. (2018). A convergence theory for deep learning via over-parameterization. *arXiv preprint arXiv:1811.03962*.

Azizan, N. and Hassibi, B. (2018). Stochastic gradient/mirror descent: Minimax optimality and implicit regularization. *arXiv preprint arXiv:1806.00952*.

Azizan, N., Lale, S., and Hassibi, B. (2019). Stochastic mirror descent on overparameterized nonlinear models: Convergence, implicit regularization, and generalization. *arXiv preprint arXiv:1906.03830*.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konecny, J., Mazzocchi, S., McMahan, H. B., et al. (2019). Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. (2018). Efficient lifelong learning with A-GEM. *arXiv preprint arXiv:1812.00420*.

De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. (2019). Continual learning: A comparative study on how to defy forgetting in classification tasks. *arXiv preprint arXiv:1909.08383*.

Díaz-Rodríguez, N., Lomonaco, V., Filliat, D., and Maltoni, D. (2018). Don't forget, there is more than forgetting: new metrics for continual learning. *arXiv preprint arXiv:1810.13166*.

Draelos, T. J., Miner, N. E., Lamb, C. C., Cox, J. A., Vineyard, C. M., Carlson, K. D., Severa, W. M., James, C. D., and Aimone, J. B. (2017). Neurogenesis deep learning: Extending deep networks to accommodate new classes. In *International Joint Conference on Neural Networks*, pages 526–533.

Ebrahimi, S., Elhoseiny, M., Darrell, T., and Rohrbach, M. (2019). Uncertainty-guided continual learning with bayesian neural networks. *arXiv preprint arXiv:1906.02425*.

Farquhar, S. and Gal, Y. (2018). Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733*.

Farquhar, S. and Gal, Y. (2019). A unifying bayesian view of continual learning. *arXiv preprint arXiv:1902.06494*.

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135.

Fusi, S., Drew, P. J., and Abbott, L. F. (2005). Cascade models of synaptically stored memories. *Neuron*, 45(4):599–611.

Gidaris, S. and Komodakis, N. (2018). Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4367–4375.

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. (2013). An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Han, B., Tsang, I. W., Xiao, X., Chen, L., Fung, S.-f., and Yu, C. P. (2018). Privacy-preserving stochastic gradual learning. *arXiv preprint arXiv:1810.00383*.

Hassabis, D., Kumaran, D., Summerfield, C., and Botvinick, M. (2017). Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258.

Hayes, T. L., Kemker, R., Cahill, N. D., and Kanan, C. (2018). New metrics and experimental paradigms for continual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2031–2034.

He, X. and Jaeger, H. (2018). Overcoming catastrophic interference using conceptor-aided backpropagation. In *ICLR 2018*.

He, X., Sygnowski, J., Galashov, A., Rusu, A. A., Teh, Y. W., and Pascanu, R. (2019). Task agnostic continual learning via meta learning. *arXiv preprint arXiv:1906.05201*.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Hsu, Y.-C., Liu, Y.-C., and Kira, Z. (2018). Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488*.

Hu, W., Lin, Z., Liu, B., Tao, C., Tao, Z., Ma, J., Zhao, D., and Yan, R. (2018). Overcoming catastrophic forgetting for continual learning via model adaptation.

Jerfel, G., Grant, E., Griffiths, T. L., and Heller, K. A. (2019). Reconciling meta-learning and continual learning with online mixtures of tasks. In *NeurIPS*.

Kamra, N., Gupta, U., and Liu, Y. (2017). Deep generative dual memory network for continual learning. *arXiv preprint arXiv:1710.10368*.

Kaplanis, C., Shanahan, M., and Clopath, C. (2018). Continual reinforcement learning with complex synapses. *arXiv preprint arXiv:1802.07239*.

Kemker, R., McClure, M., Abitino, A., Hayes, T. L., and Kanan, C. (2018). Measuring catastrophic forgetting in neural networks. In *Thirty-second AAAI conference on artificial intelligence*.

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.

Kolouri, S., Ketz, N., Zou, X., Krichmar, J., and Pilly, P. (2019). Attention-based structural-plasticity. *arXiv preprint arXiv:1903.06070*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kumaran, D. and McClelland, J. L. (2012). Generalization through the recurrent interaction of episodic memories: a model of the hippocampal system. *Psychological review*, 119(3):573.

Lee, S.-W., Kim, J.-H., Jun, J., Ha, J.-W., and Zhang, B.-T. (2017). Overcoming catastrophic forgetting by incremental moment matching. In *Neural information processing systems*, pages 4652–4662.

Li, A., Hu, H., Mirowski, P., and Farajtabar, M. (2019a). Cross-view policy learning for street navigation. *arXiv preprint arXiv:1906.05930*.

Li, C., Zhou, P., Xiong, L., Wang, Q., and Wang, T. (2018). Differentially private distributed online learning. *IEEE Transactions on Knowledge and Data Engineering*, 30(8):1440–1453.

Li, X., Zhou, Y., Wu, T., Socher, R., and Xiong, C. (2019b). Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. *arXiv preprint arXiv:1904.00310*.

Li, Y. and Liang, Y. (2018). Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, pages 8157–8166.

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

Lopez-Paz, D. and Ranzato, M. (2017). Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476.

Lüders, B., Schläger, M., and Risi, S. (2016). Continual learning through evolvable neural turing machines. In *NIPS 2016 Workshop on Continual Learning and Deep Networks (CLDL 2016)*.

McCloskey, M. and Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Mirzadeh, S.-I., Farajtabar, M., Li, A., and Ghasemzadeh, H. (2019). Improved knowledge distillation via teacher assistant: Bridging the gap between student and teacher. *arXiv preprint arXiv:1902.03393*.

Nguyen, C. V., Achille, A., Lam, M., Hassner, T., Mahadevan, V., and Soatto, S. (2019). Toward understanding catastrophic forgetting in continual learning. *arXiv preprint arXiv:1908.01091*.

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

Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural Networks*.

Pihur, V., Korolova, A., Liu, F., Sankuratripati, S., Yung, M., Huang, D., and Zeng, R. (2018). Differentially-private" draw and discard" machine learning. *arXiv preprint arXiv:1807.04369*.

Ratcliff, R. (1990). Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285.

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., and Tesauro, G. (2018). Learning

to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*.

Rios, A. and Itti, L. (2018). Closed-loop gan for continual learning. *arXiv preprint arXiv:1811.01146*.

Ritter, H., Botev, A., and Barber, D. (2018). Online structured laplace approximations for overcoming catastrophic forgetting. In *Advances in Neural Information Processing Systems*, pages 3738–3748.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T. P., and Wayne, G. (2018). Experience replay for continual learning. *arXiv preprint arXiv:1811.11682*.

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. (2014). Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., and Hadsell, R. (2018). Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pages 4535–4544.

Shin, H., Lee, J. K., Kim, J., and Kim, J. (2017). Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Smith, V., Chiang, C.-K., Sanjabi, M., and Talwalkar, A. S. (2017). Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4424–4434.

Sorokin, A. Y. and Burtsev, M. S. (2019). Continual and multi-task reinforcement learning with shared episodic memory. *arXiv preprint arXiv:1905.02662*.

Srivastava, R. K., Masci, J., Kazerounian, S., Gomez, F., and Schmidhuber, J. (2013). Compete to compute. In *Advances in neural information processing systems*, pages 2310–2318.

Thrun, S. and Mitchell, T. M. (1995). Lifelong robot learning. *Robotics and autonomous systems*, 15(1-2):25–46.

Titsias, M. K., Schwarz, J., Matthews, A. G. d. G., Pascanu, R., and Teh, Y. W. (2019). Functional regularisation for continual learning using gaussian processes. *arXiv preprint arXiv:1901.11356*.

Toneva, M., Sordoni, A., Combes, R. T. d., Trischler, A., Bengio, Y., and Gordon, G. J. (2018). An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*.

Vepakomma, P., Gupta, O., Swedish, T., and Raskar, R. (2018). Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*.

Vuorio, R., Cho, D.-Y., Kim, D., and Kim, J. (2018). Meta continual learning. *arXiv preprint arXiv:1806.06928*.

Wen, J., Cao, Y., and Huang, R. (2018). Few-shot self reminder to overcome catastrophic forgetting. *arXiv preprint arXiv:1812.00543*.

Wu, X., Li, F., Kumar, A., Chaudhuri, K., Jha, S., and Naughton, J. (2017). Bolt-on differential privacy for scalable stochastic gradient descent-based analytics. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1307–1322. ACM.

Xiao, T., Zhang, J., Yang, K., Peng, Y., and Zhang, Z. (2014). Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 177–186. ACM.

Yoon, J., Yang, E., Lee, J., and Hwang, S. J. (2018). Lifelong learning with dynamically expandable networks. In *Sixth International Conference on Learning Representations*. ICLR.

Zeng, G., Chen, Y., Cui, B., and Yu, S. (2018). Continuous learning of context-dependent processing in neural networks. *arXiv preprint arXiv:1810.01256*.

Zenke, F., Poole, B., and Ganguli, S. (2017). Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3987–3995. JMLR.

Zhang, M., Wang, T., Lim, J. H., and Feng, J. (2019). Prototype reminding for continual learning. *arXiv preprint arXiv:1905.09447*.

A Additional Experiment Information

A.1 Variants of Orthogonal Gradient Descent

As described in Section 3 we test OGD with three settings for the which gradients to store, and 3 settings for how many gradients to store. OGD-ALL stores the gradients with respect to all logits of the model. OGD-AVG stores the gradients with respect to the average of all logits. OGD-GTL stores the gradient with respect to the ground truth logit. We run tests storing 20, 200, and 2000 gradients. Table A.1 summarizes the results of this experiment. We observe that increasing the number of gradients improves performance across the board (which is expected). We observe the OGD-GTL and OGD-ALL have similar performance in most cases, with a bit of an edge to OGD-GTL. OGD-AVG performs worse in most cases.

rotated Mnist	Accuracy ± Std (%)		
	20 Grads	200 Grads	2000 Grads
all	75.7 ± 2.6	79.9 ± 1.4	86.6 ± 1.0
average	75.3 ± 2.4	75.5 ± 1.4	77.7 ± 1.6
ground truth	76.4 ± 2.2	82.9 ± 1.6	87.1 ± 1.1
permuted Mnist	**Accuracy ± Std (%)**		
	20 Grads	200 Grads	2000 Grads
all	87.3 ± 2.8	89.7 ± 1.5	90.5 ± 0.9
average	86.8 ± 1.4	86.9 ± 1.4	89.4 ± 1.7
ground truth	86.5 ± 1.5	89.4 ± 1.0	91.4 ± 1.7

Table 4: The performance of various OGD gradient methods as a function of number of gradients stored on rotated Mnist (top) and permuted Mnist (bottom). Numbers are the accuracy on task 1 after fully training on task 2.

A.2 Increased Training Epochs

We study the effect that increasing the number of training epochs has on the performance of the different training methods on permuted Mnist. For the Mnist experiments in the Section 4, we train for 5 epochs per task, which is enough to achieve 93% accuracy on vanilla Mnist classification and is in the regime short enough to avoid over-fitting. In order to determine whether increased training time has an effect on the performance in the multi-task setting, we train a classifier on 2-task permuted Mnist running each task training for 20, 40, 80, and 120 epochs and report the classification accuracy on task 1 after task 2 has finished. The results are shown in Figure 5. Note that A-GEM and OGD have maintained competitive performance with increasing number of epochs while in the case of SGD and EWC the performance first increases and then drops.

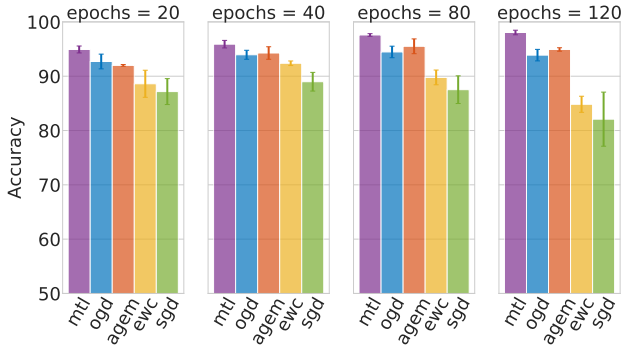

Figure 5: The performance of OGD versus others as a function of the number of training epochs for each task on permuted Mnist.

A.3 Split Mnist

We present the results of the split Mnist study described in Section 4.3 on 3 other instances of the split Mnist task. These instances differ by the way the Mnist classes are split into tasks and the order in which the tasks are presented. Tables 5, 6, and 7 show the results on these tests. We can see that the ordering of the task methods is preserved in all tests: MTL and OGD are very close in performance, with a gap before A-GEM, and finally EWC and SGD.

	Accuracy ± Std. (%)				
	Task 1	Task 2	Task 3	Task 4	Task 5
mtl	99.6 ± 0.2	98.5 ± 0.4	97.7 ± 0.4	96.8 ± 1.0	98.7 ± 0.3
ogd	99.6 ± 0.4	97.7 ± 0.1	97.3 ± 0.5	98.0 ± 0.9	99.3 ± 0.1
agem	99.2 ± 0.6	91.4 ± 3.7	91.4 ± 0.9	87.1 ± 3.9	98.9 ± 0.3
ewc	97.0 ± 3.2	92.7 ± 3.8	91.9 ± 5.7	94.3 ± 2.2	99.2 ± 0.6
sgd	97.4 ± 2.4	92.2 ± 3.5	89.2 ± 8.6	94.5 ± 1.4	99.1 ± 0.3

Table 5: The accuracy of models trained by different methods on split Mnist. The reported values are the accuracy of the model for test examples from the indicated class after the model has been trained on all tasks in sequence. This table contains the same settings as Table 3, but with a different order of Mnist classes assigned to the tasks.

	Accuracy ± Std. (%)				
	Task 1	Task 2	Task 3	Task 4	Task 5
mtl	99.4 ± 0.2	99.2 ± 0.3	98.6 ± 0.4	99.7 ± 0.3	98.6 ± 0.5
ogd	99.0 ± 0.4	98.6 ± 0.1	98.0 ± 0.2	99.6 ± 0.3	99.6 ± 0.2
agem	94.1 ± 2.9	93.8 ± 5.5	90.6 ± 2.2	99.4 ± 0.3	99.4 ± 0.3
ewc	94.8 ± 2.9	95.3 ± 3.1	95.5 ± 0.6	99.3 ± 0.2	99.3 ± 0.2
sgd	94.6 ± 2.1	96.3 ± 1.2	95.0 ± 1.6	99.3 ± 0.4	99.3 ± 0.2

Table 6: The accuracy of models trained by different methods on split Mnist. The reported values are the accuracy of the model for test examples from the indicated class after the model has been trained on all tasks in sequence. This table contains the same settings as Table 3, but with a different order of Mnist classes assigned to the tasks.

	Accuracy ± Std. (%)				
	Task 1	Task 2	Task 3	Task 4	Task 5
mtl	98.4 ± 0.2	100.0 ± 0.0	98.6 ± 0.3	99.5 ± 0.2	98.9 ± 0.5
ogd	98.1 ± 0.8	99.9 ± 0.1	97.8 ± 0.6	99.4 ± 0.3	99.5 ± 0.3
agem	92.1 ± 2.7	93.8 ± 8.2	93.0 ± 3.5	98.6 ± 0.5	99.5 ± 0.3
ewc	92.5 ± 2.2	98.1 ± 3.0	94.0 ± 0.9	99.4 ± 0.2	99.5 ± 0.3
sgd	89.6 ± 4.4	98.9 ± 1.0	89.1 ± 7.9	98.9 ± 0.7	99.5 ± 0.3

Table 7: The accuracy of models trained by different methods on split Mnist. The reported values are the accuracy of the model for test examples from the indicated class after the model has been trained on all tasks in sequence. This table contains the same settings as Table 3, but with a different order of Mnist classes assigned to the tasks.