Rotate your Networks: Better Weight Consolidation and Less Catastrophic Forgetting

Xialei Liu*, Marc Masana*, Luis Herranz, Joost Van de Weijer, Antonio M. López
Computer Vision Center, UAB
Barcelona, Spain
Email: {xialei, mmasana, lherranz, joost, antonio}@cvc.uab.es

Andrew D. Bagdanov
MICC, University of Florence
Florence, Italy
Email: andrew.bagdanov@unifi.it

Abstract—**In this paper we propose an approach to avoiding catastrophic forgetting in sequential task learning scenarios. Our technique is based on a network reparameterization that approximately diagonalizes the Fisher Information Matrix of the network parameters. This reparameterization takes the form of a factorized rotation of parameter space which, when used in conjunction with Elastic Weight Consolidation (which assumes a diagonal Fisher Information Matrix), leads to significantly better performance on lifelong learning of sequential tasks. Experimental results on the MNIST, CIFAR-100, CUB-200 and Stanford-40 datasets demonstrate that we significantly improve the results of standard elastic weight consolidation, and that we obtain competitive results when compared to the state-of-the-art in lifelong learning without forgetting.**

I. INTRODUCTION

Neural networks are very effective models for a variety of computer vision tasks. In general, during training these networks are presented with examples from all tasks they are expected to perform. In a lifelong learning setting, however, learning is considered as a sequence of tasks to be learned [1], which is more similar to how biological systems learn in the real world. In this case networks are presented with *groups* of tasks, and at any given moment the network has access to training data from only *one* group. The main problem which systems face in such settings is *catastrophic forgetting*: while adapting network weights to new tasks, the network forgets the previously learned ones [2].

There are roughly two main approaches to lifelong learning (which has seen increased interest in recent years). The first group of methods stores a small subset of training data from previously learned tasks. These stored exemplars are then used during training of new tasks to avoid forgetting the previous ones [3], [4]. The second type of algorithm instead avoids storing any training data from previously learned tasks. A number of algorithms in this class are based on Elastic Weight Consolidation (EWC) [5]–[7], which includes a regularization term that forces parameters of the network to remain close to the parameters of the network trained for the previous tasks. In a similar vein, Learning Without Forgetting (LWF) [8] regularizes *predictions* rather than weights. Aljundi et al. [9] learns a representation for each task and use a set of gating autoencoders to decide which expert to use at testing time.
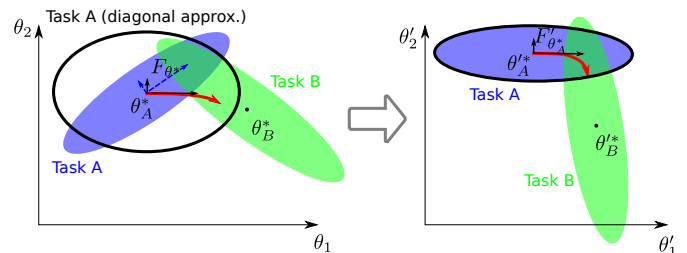

Fig. 1: Sequential learning in parameter space, illustrating the optimal model parameters for tasks A and B and regions with low forgetting. The red line indicates the learning path for task B from the previously learned solution for task A. **Left**: the diagonal approximation (black ellipse) of the Fisher Information Matrix can be poor and steer EWC in directions that will forget task A. **Right**: after a suitable reparameterization (i.e., a rotation) the diagonal approximation is better and EWC can avoid forgetting task A.

EWC is an elegant approach to selective regularization of network parameters when switching tasks. It uses the Fisher Information Matrix (FIM) to identify directions in feature space critical to performing already learned tasks (and as a consequence also those directions in which the parameters may move freely without forgetting learned tasks). However, EWC has the drawback that it assumes the Fisher Information Matrix to be diagonal – a condition that is almost never true.

In this paper we specifically address this diagonal assumption made by the EWC algorithm. If the FIM is not diagonal, EWC can fail to prevent the network from straying away from "good parameter space" (see Fig. 1, left). Our method is based on *rotating* the parameter space of the neural network in such a way that the output of the forward pass is unchanged, but the FIM computed from gradients during the backward pass is approximately diagonal (see Fig. 1, right). The result is that EWC in this rotated parameter space is significantly more effective at preventing catastrophic forgetting in sequential task learning problems. An extensive experimental evaluation on a variety of sequential learning tasks shows that our approach significantly outperforms standard elastic weight consolidation.

The rest of the paper is organized as follows. In the next section we review the EWC learning algorithm and in Sec-

*Both authors contributed equally.

tion IV we describe the approach to rotating parameter space in order to better satisfy the diagonal requirement of EWC. We give an extensive experimental evaluation of the proposed approach in Section V and conclude with a discussion of our contribution in Section VI.

II. RELATED WORK

An intuitive approach to avoiding forgetting for sequential task learning is to retain a portion of training data for each task. The iCaRL method of Rebuffi et al. [3] is based on retaining exemplars which are rehearsed during the training of new tasks. Their approach also includes a method for exemplar herding which ensures that the class mean remains close even when the number of exemplars per class is dynamically varied. A recent paper proposed the Gradient Episodic Memory model [4]. Its main feature is an episodic memory storing a subset of the observed examples. However, these examples are not directly rehearsed but instead are used to define inequality constraints on the loss that ensure that it does not increase with respect to previous tasks. A cooperative dual model architecture consisting of a deep generative model (from which training data can be sampled) and a task solving model is proposed in [10].

Learning without Forgetting in [8] works without retaining training data from previous tasks by regularizing the network output on new task data to not stray far from its original output. Similarly, the lifelong learning approach described in [11] identifies informative features using per-task autoencoders and then regularizes the network to preserve these features in its internal, shared-task feature representation when training on a new task. Elastic Weight Consolidation (EWC) [5]–[7] includes a regularization term that forces important parameters of the network to remain close to the parameters of the network trained for the previous tasks. The authors of [12] propose a task-based hard attention mechanism that preserves information about previous tasks without affecting the current task's learning. Aljundi et al. [9] learn a representation for each task and use a set of gating autoencoders to decide which expert to use at testing time.

Another class of learning methods related to our approach are those based on the *Natural Gradient Descent (NGD)* [13], [14]. The NGD uses the Fisher Information Matrix (FIM) as the natural metric in parameter space. Candidate directions are projected using the inverse FIM and the best step (in terms of decreasing loss) is taken. The Projected NGD algorithm estimates a network reparamaterization online that whitens the FIM so that vanilla Stochastic Gradient Descent (SGD) is equivalent to NGD [15]. The authors of [16] show how the natural gradient can be used to explain and classify a range of adaptive stepsize strategies for training networks. In [17] the authors propose an approximation to the FIM for convolutional networks and show that the resulting NGD training procedure is many times more efficient than SGD.

III. ELASTIC WEIGHT CONSOLIDATION

Elastic Weight Consolidation (EWC) addresses the problem of catastrophic forgetting in continual and sequential task learning in neural networks [2], [5]. In this section we give a brief overview of EWC and discuss some of its limitations.

A. Overview

The problem addressed by EWC is that of learning the K-th task in a network that already has learned $K-1$ tasks. The main challenge is to learn the new task in a way that prevents *catastrophic forgetting*. Catastrophic forgetting occurs when new learning interferes catastrophically with prior learning during sequential neural network training, which causes the network to partially or completely forget previous tasks [2].

The final objective is to learn the optimal set of parameters $\theta^*_{1:K}$ of a network given the previous ones $\theta^*_{1:K-1}$ learned from the previous $K-1$ tasks.[1] Each of the K tasks consists of a training dataset $\mathcal{D}_k = (\mathcal{X}_k, \mathcal{Y}_k)$ with samples $x \in \mathcal{X}_k$ and labels $y \in \mathcal{Y}_k$ (and for previous tasks $\mathcal{D}_{1:K-1} = (\mathcal{X}_1, \ldots, \mathcal{X}_{K-1}, \mathcal{Y}_1, \ldots, \mathcal{Y}_{K-1})$). We are interested in the configuration θ which maximizes the posterior $p_{1:K} \equiv p(\theta|\mathcal{D}_{1:K})$.

EWC [5] uses sequential Bayesian estimation [6] to factorize $p_{1:K}$ as:

$$\log p_{1:K} = \log p(\mathcal{Y}_K|\mathcal{X}_K; \theta) + \log p(\theta|\mathcal{D}_{1:K-1}) \quad (1)$$
$$- \log p(\mathcal{D}_K|\mathcal{D}_{1:K-1}) + C,$$

where $p_{1:K-1}$ is the prior model from previous tasks, $p_K \equiv p(\theta|\mathcal{X}_K, \mathcal{Y}_K)$ is the posterior probability corresponding to the new task, and C is a constant factor that can be ignored for training.

Since calculating the true posterior is intractable, EWC uses the Laplace approximation to approximate the posterior with a Gaussian:

$$\log p_{1:K} \approx \log p(\mathcal{Y}_K|\mathcal{X}_K; \theta) \quad (2)$$
$$- \frac{\lambda}{2}(\theta - \theta^*_{1:K-1})^\mathsf{T} \tilde{F}_{1:K-1}(\theta - \theta^*_{1:K-1}) + C'$$
$$\approx \log p(\mathcal{Y}_K|\mathcal{X}_K; \theta) \quad (3)$$
$$- \frac{\lambda}{2} \sum_i \tilde{F}_{1:K-1,ii}(\theta_i - \theta^*_{1:K-1,i})^2 + C',$$

where $\tilde{F}_{1:K-1}$ is the Fisher Information Matrix (FIM) that approximates the inverse of the covariance matrix at $\theta^*_{1:K-1}$ used in the Laplace approximation of $\log p_{1:l}$ in (2), and the second approximation in (3) assumes $\tilde{F}$ is diagonal – a common assumption in practice – and thus that the quadratic form in (2) can be replaced with scaling by the diagonal entries $\tilde{F}_{1:K-1,ii}$ of the FIM at $\theta^*_{1:K-1}$.

The FIM of the true distribution $p(y|x; \theta)$ is estimated as:

$$F_\theta = \mathbb{E}_{x \sim \pi}\left\{\mathbb{E}_{y \sim p(y|x;\theta)}\left[\left(\frac{\partial \log p}{\partial \theta}\right)\left(\frac{\partial \log p}{\partial \theta}\right)^\mathsf{T}\right]\right\} \quad (4)$$
$$= \mathbb{E}_{x \sim \pi}\left\{\mathbb{E}_{y \sim p(y|x;\theta)}\left[\frac{\partial^2 \log p}{\partial \theta^2}\right]\right\}, \quad (5)$$

[1]We adapt the notation from [6] and [15].

where π is the empirical distribution of a training set $\mathcal{X}$.

This definition of the FIM as the second derivatives of the log-probability is key to understanding its role in preventing forgetting. Once a network is trained to a configuration θ^*, the FIM F_{θ^*} indicates how prone each dimension in the parameter space is to causing forgetting when gradient descent updates the model to learn a new task: it is preferable to move along directions with low Fisher information, since $\log p$ decreases slowly (the red line in Fig. 1). EWC uses F_{θ^*} in the regularization term of (2) and (3) to penalize moving in directions with higher Fisher information and which are thus likely to result in forgetting of already-learned tasks. EWC uses different regularization terms per task [5]. Instead of using (3), we only use the FIM from the previous task because we found it works better, requires storage of only one FIM, and better fits the sequential Bayesian perspective (as shown in [18]).

B. Limitations of EWC

Assuming the FIM to be diagonal is a common practice in the Laplace approximation for two reasons. First, the number of parameters is reduced from $O(N^2)$ to $O(N)$, where N is the number of elements in parameter space θ, so a diagonal matrix is much more efficient to compute and store. Additionally, in many cases the required matrix is the inverse of the FIM (for example in NGD methods [13]), which is significantly simpler and faster to compute for diagonal matrices.

However, in the case of sequential learning of new tasks with EWC, the diagonal FIM assumption might be unrealistic – at least in the original parameter space. On the left of Fig. 1 is illustrated a situation where a simple Gaussian distribution (solid blue ellipse) is approximated using a diagonal covariance matrix (black ellipse). By rotating the parameter space so that $\frac{\partial \log p}{\partial \theta}$ are aligned (on average) with the coordinate axes (Fig. 1, right), the diagonal assumption is more reasonable (or even true in the case of a Gaussian distribution). In this rotated parameter space, EWC is better able to optimize the new task while not forgetting the old one.

The example in Fig. 2a shows the FIM obtained for the weights in the second layer of a multilayer perceptron trained on MNIST [19] (specifically, four dense layers with 784, 10, 10, and 10 neurons). The matrix is clearly non-diagonal, so the diagonal approximation misses significant correlations between weights and this may lead to forgetting when using the diagonal approximation. The diagonal only retains 40.8% of the energy of the full matrix in this example.

IV. ROTATED ELASTIC WEIGHT CONSOLIDATION

Motivated by the previous observation, we aim to find a reparameterization of the parameter space θ. Specifically, we desire a reparameterization which does not change the feed-forward response of the network, but that better satisfies the assumption of diagonal FIM. After reparameterization, we can assume a diagonal FIM which is efficiently estimated in
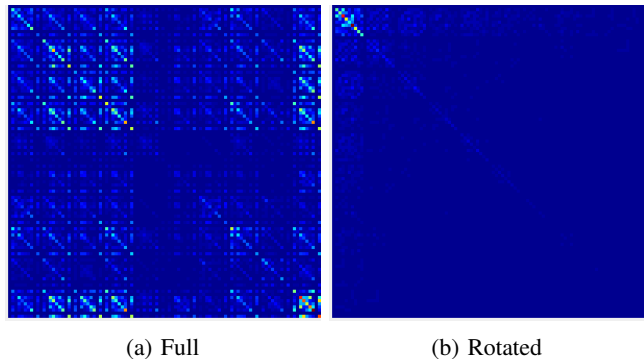

(a) Full   (b) Rotated

Fig. 2: Fisher Information Matrix: (a) original and (b) rotated using the proposed technique. The range is color coded and normalized for better visualization.

that new parameter space. Finally, minimization by gradient descent on the new task is also performed in this new space.

One possible way to obtain this reparameterization is by computing a rotation matrix using the Singular Value Decomposition (SVD) of (4). Note that this decomposition is performed in the parameter space. Unfortunately, this approach has three problems. First, the SVD is extremely expensive to compute on very large matrices. Second, this rotation ignores the sequential structure of the neural network and would likely catastrophically change the feed-forward behavior of the network. Finally, we do not have the FIM in the first place.

A. Indirect rotation

In this section we show how rotation of fully connected and convolutional layer parameters can be applied to obtain a network for which the assumption of diagonal FIM is more valid. These rotations are chosen so as to not alter the feed-forward response of the network.

For simplicity, we first consider the case of a single fully-connected layer given by the linear model $\mathbf{y} = W\mathbf{x}$, with input $\mathbf{x} \in \mathbb{R}^{d_1}$, output $\mathbf{y} \in \mathbb{R}^{d_2}$ and weight matrix $W \in \mathbb{R}^{d_2 \times d_1}$. In this case $\theta = W$, and to simplify the notation we use $L = \log p(\mathbf{y}|\mathbf{x}; W)$. Using (4), the FIM in this simple linear case is (after applying the chain rule):

$$
\begin{aligned}
F_W &= \mathbb{E}_{\substack{x \sim \pi \\ y \sim p}} \left[\left(\frac{\partial L}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial W} \right) \left(\frac{\partial L}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial W} \right)^\mathsf{T} \right] \\
&= \mathbb{E}_{p \sim \pi} \left[\left(\frac{\partial L}{\partial \mathbf{y}} \right) \mathbf{x}\mathbf{x}^\mathsf{T} \left(\frac{\partial L}{\partial \mathbf{y}} \right)^\mathsf{T} \right].
\end{aligned} \tag{6}
$$

If we assume that $\frac{\partial L}{\partial \mathbf{y}}$ and $\mathbf{x}$ are independent random variables we can factorize (6) as done in [15]:

$$
F_W = \mathbb{E}_{\substack{x \sim \pi \\ y \sim p}} \left[\left(\frac{\partial L}{\partial \mathbf{y}} \right) \left(\frac{\partial L}{\partial \mathbf{y}} \right)^\mathsf{T} \right] \mathbb{E}_{x \sim \pi} \left[\mathbf{x}\mathbf{x}^\mathsf{T} \right], \tag{7}
$$

which indicates that we can approximate the FIM using two independent factors that only depend on the backpropagated gradient at the output $\frac{\partial L}{\partial \mathbf{y}}$ and on the input $\mathbf{x}$, respectively. This result also suggests that there may exist a pair of rotations of
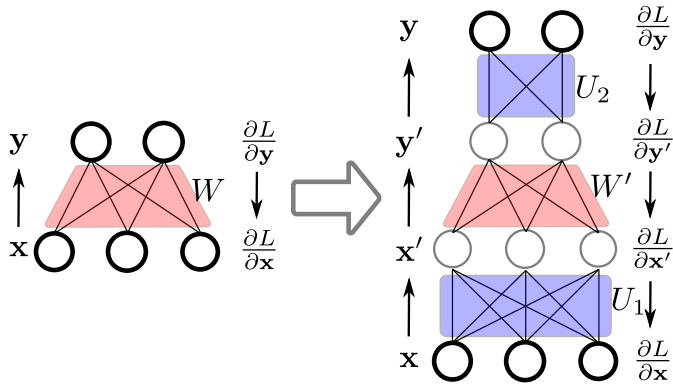
Fig. 3: Reparameterization of a linear layer W as W' using two additional linear layers U_1 and U_2.

Fig. 4: Reparameterization of a convolutional layer K as K' using two additional 1×1 convolutional layers U_1 and U_2 as rotations.

the input and the output, respectively, that lead to a rotation of F_W in the parameter space W.

In fact, these rotation matrices can be obtained as $U_1 \in \mathbb{R}^{d_1 \times d_1}$ and $U_2 \in \mathbb{R}^{d_2 \times d_2}$ from the following SVD decompositions:

$$\mathbb{E}_{x \sim \pi}\left[\mathbf{x}\mathbf{x}^{\mathsf{T}}\right] = U_1 S_1 V_1^{\mathsf{T}} \qquad (8)$$

$$\mathbb{E}_{\substack{x \sim \pi \\ y \sim p}}\left[\left(\frac{\partial L}{\partial \mathbf{y}}\right)\left(\frac{\partial L}{\partial \mathbf{y}}\right)^{\mathsf{T}}\right] = U_2 S_2 V_2^{\mathsf{T}} \qquad (9)$$

Since both rotations are local, i.e. they are applied to a single layer, they can be integrated in the network architecture as two additional (fixed) linear layers $\mathbf{x}' = U_1\mathbf{x}$ and $\mathbf{y} = U_2\mathbf{y}'$ (see Fig. 3).

The new, rotated weight matrix is then:

$$W' = U_2^{\mathsf{T}} W U_1^{\mathsf{T}}. \qquad (10)$$

Thus, that the forward passes of both networks in Fig. 3 is equivalent since $U_2 W' U_1 = W$. In this way, the sequential structure of the network is not broken, and the learning problem is equivalent to learning the parameters of W' for the new problem $\mathbf{y}' = W'\mathbf{x}'$. The use of layer decomposition with SVD was also investigated in the context of network compression [20], [21]. However this SVD analysis was based on layer weight matrices and the original network layer was only decomposed into two new layers.

The training procedure is exactly the same as in (2), but using $\mathbf{x}'$, $\mathbf{y}'$ and W' instead of $\mathbf{x}$, $\mathbf{y}$ and W to estimate the FIM and to learn the weights. The main difference is that the approximate diagonalization of W in W' will be more effective in preventing forgetting using EWC. Fig. 2b shows the resulting matrix after applying the proposed rotations in the previous example. Note that most of the energy concentrates in fewer weights and it is also better conditioned for a diagonal approximation (in this case the diagonal retains 74.4% of the energy).

Assuming a block-diagonal FIM, the extension to multiple layers is straightforward by applying the same procedure layer-wise using the corresponding inputs instead of $\mathbf{x}$ and backpropagated gradients to the output of the layer as $\frac{\partial L}{\partial \mathbf{y}}$
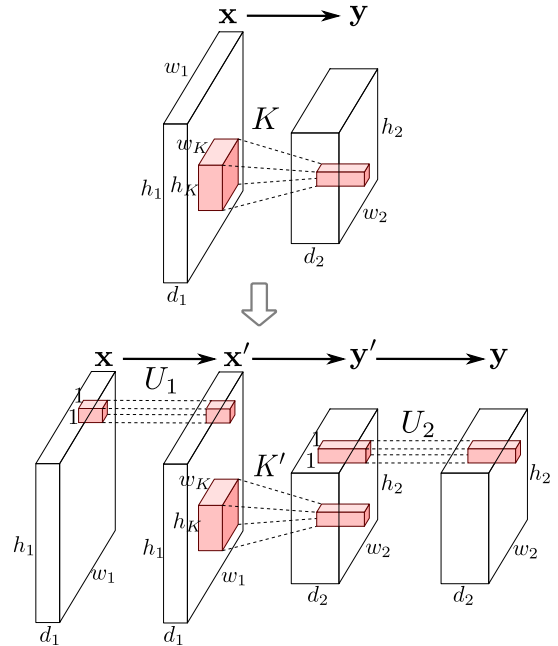
(that is, estimating the FIM for each layer, and computing layer-specific U_1, W' and U_2). In Algorithm 1 we describe the reparameterization used in the training process.

B. Extension to convolutional layers

The method proposed in the previous section for fully connected layers can be applied to convolutional layers with only slight modifications. The rotations are performed by adding two additional 1×1 convolutional layers (see Fig. 4).

Assume we have an input tensor $\mathbf{x} \in \mathbb{R}^{w_1 \times h_1 \times d_1}$, a kernel tensor $K \in \mathbb{R}^{w_k \times h_k \times d_1 \times d_2}$, and that the corresponding output tensor is $\mathbf{y} \in \mathbb{R}^{w_2 \times h_2 \times d_2}$. For convenience let the mode-3 fiber[2] of $\mathbf{x}$ be $\mathbf{x}_{l,m} = (x_{l,m,i}|i = 1, \ldots, d_1)^{\mathsf{T}}$ and of the output gradient tensor $\frac{\partial L}{\partial \mathbf{y}}$ as $\mathbf{z}_{l,m} = \left(\left(\frac{\partial L}{\partial \mathbf{y}}\right)_{l,m,i}|i = 1, \ldots, d_2\right)^{\mathsf{T}}$. Note that each $\mathbf{x}_{l,m}$ and $\mathbf{z}_{l,m}$ are d_1-dimensional and d_2-dimensional vectors, respectively. Now we can compute the self-correlation matrices averaged over all spatial coordinates as:

$$X = \frac{1}{w_1 h_1} \sum_{l=1}^{w_1} \sum_{m=1}^{h_1} \mathbf{x}_{l,m}\mathbf{x}_{l,m}^{\mathsf{T}} \qquad (11)$$

$$Z = \frac{1}{w_2 h_2} \sum_{l=1}^{w_2} \sum_{m=1}^{h_2} \mathbf{z}_{l,m}\mathbf{z}_{l,m}^{\mathsf{T}}, \qquad (12)$$

---

[2]A mode-i fiber of a tensor is defined as the vector obtained by fixing all its indices but i. A slice of a tensor is a matrix obtained by fixing all its indices but two. See [22] for more details.

Algorithm 1 : Incremental task learning.

Input: $\mathcal{D}_k = (\mathcal{X}_k, \mathcal{Y}_k)$, training samples in per-class sets.
Require: Initial parameters Θ_0, total number of tasks K
for $k = 1, \ldots, K$
    if $k=1$
        $\Theta \leftarrow \text{UPDATE}((\mathcal{X}_k, \mathcal{Y}_k), \Theta_0, 0)$
    else
        $F_\Theta \leftarrow \text{COMPUTE\_FIM}((\mathcal{X}_{k-1}, \mathcal{Y}_{k-1}), \Theta^R)$
        $\Theta^R \leftarrow \text{UPDATE}((\mathcal{X}_k, \mathcal{Y}_k), \Theta^R, F_\Theta)$
        $\Theta \;\; \leftarrow \text{COMBINE}(\Theta^R)$
    $\Theta^R \leftarrow \text{ROTATE}(\Theta)$
end for

UPDATE$((\mathcal{X}_k, \mathcal{Y}_k), \Theta^R, F)$ above fits the model for task k using EWC, with rotated parameters Θ^R, and FIM F (0 is the zero matrix). COMBINE(Θ^R) fuses current parameters before computation of new rotated parameters for the coming tasks.

---

and compute the decompositions of (8) and (9) as

$$\mathbb{E}_{x \sim \pi}[X] \;\; = \;\; U_1 S_1 V_1^\mathsf{T} \tag{13}$$

$$\mathbb{E}_{\substack{x \sim \pi \\ y \sim p}}[Z] \;\; = \;\; U_2 S_2 V_2^\mathsf{T}. \tag{14}$$

We define $K_{l,m} = (k_{l,m,i,j} | i = 1, \ldots, d_1, j = 1, \ldots, d_2)$, which is a slice² of the kernel tensor K. The rotated slices are then obtained as:

$$K'_{l,m} = U_2^\mathsf{T} K_{l,m} U_1^\mathsf{T}. \tag{15}$$

and the final rotated kernel tensor $K' \in \mathbb{R}^{w_k \times h_k \times d_1 \times d_2}$ is obtained simply by tiling all slices $K'_{l,m}$ computed for every l and m.

V. EXPERIMENTAL RESULTS

In this section we report on a number of experiments comparing our approach to EWC [5] and other baselines.³

A. Experimental settings

Datasets. We evaluate our method on two small datasets and three fine-grained classification datasets: MNIST [19], CIFAR-100 [23], CUB-200 Birds [24] and Stanford-40 Actions [25]. Each dataset is equally divided into T groups of classes, which are seen as the sequential tasks to learn. In the case of the CUB-200 dataset, we crop the bounding boxes from the original images and resize them to 224×224. For Actions, we resize the input images to 256×256, then take random crops of size 224×224. We perform no data augmentation for the MNIST and CIFAR-100 datasets.

Training details. We chose the LeNet [26] and VGG-16 [27] network architectures, which are slightly modified due to the requirements of different datasets. For MNIST, we add 2×2 padding to the original 28×28 images to obtain 32×32 input images for LeNet. LeNet is trained from scratch, while for CIFAR-100 the images are passed through the VGG-16 [27]

³Code available at: https://github.com/xialeiliu/RotateNetworks

network pre-trained on ImageNet, which has been shown to perform well when changing to other domains. The input images are $32 \times 32 \times 3$ and this provides a feature vector of $1 \times 1 \times 512$ at the end of the *pool5* layer. We use those feature vectors as an input to a classification network consisting of 3 fully-connected layers of output dimensions of 256, 256 and 100, respectively. For the two fine-grained datasets, we fine-tune from the pre-trained model on ImageNet. To save memory and limit computational complexity, we add a global pooling layer after the final convolutional layer of VGG-16. The fully-connected layers used for our experiments are of size 512, 512 and the size of the output layer corresponding to the number of classes in each dataset. The Adam optimizer is used with a learning rate of 0.001 for all experiments. We train for 5 epochs on MNIST and for 50 epochs on the other datasets.

Evaluation protocols. In our experiments, we share all layers in the networks across all tasks during training, which allows us to perform inference without explicitly knowing the task. We report the classification accuracy of each previous task $T-1$ and the current task T after training on the T-th task. When the number of tasks is large, we only report the average classification accuracy over all trained tasks.

Lifelong learning is evaluated in two settings depending on knowledge of the task label at inference time. Here we consider the more difficult scenario where task labels are unknown, and results cannot be directly compared to methods which consider labels [3], [4], [10]. As a consequence our method is implemented with as the last layer in a single network head, which is also used in [28]. During training we increase the number of output neurons as new tasks are added.

B. Disjoint MNIST comparison and ablation study

We use the disjoint MNIST dataset [19], which assigns half of the numbers as task 1 and the rest as task 2. We compare our method (R-EWC) with fine-tuning (FT) and Elastic Weight Consolidation (EWC) [5]. First, task 1 is learned from scratch on the LeNet architecture. For FT, task 2 is learned starting from task 1 as initialization. For EWC and R-EWC, task 2 is learned according to the corresponding method. In addition, we also train task 2 with only applying R-EWC to the convolutional layers (conv only), to fully connected layers (fc only), and to all layers except for the last fully-connected (all no last).

Table I compares the performance of the proposed methods for different values for the trade-off parameter λ between classification loss and FIM regularization. Results were obtained using 200 randomly selected samples (40 per class) from the validation set for computing the FIM. Each experiment was executed 3 times and the results in Table I are the average.

Results show that R-EWC clearly outperforms FT and EWC for all λ, while the best trade-off value might vary depending on the layers involved. As expected, lower values of the trade-off tend towards a more FT strategy where task 1 is forgotten more quickly. On the other hand, larger values of the trade-off give more importance to keeping the weights useful for task 1, avoiding catastrophic forgetting but also making task

TABLE I: Ablation study on Disjoint MNIST when $T=2$. Numbers in **bold** indicate the best performing configuration of each method. All versions of R-EWC that rotate fully-connected layers significantly outperform EWC.

	$\lambda = 1$		$\lambda = 10$		$\lambda = 100$		$\lambda = 1000$		$\lambda = 10000$	
	Task 1	Task 2	Task 1	Task 2	Task 1	Task 2	Task 1	Task 2	Task 1	Task 2
FT	6.1	97.6	6.1	97.6	6.1	97.6	6.1	97.6	6.1	97.6
EWC [5]	66.8	90.9	75.3	95.6	**85.8**	**92.8**	78.4	93.7	81.0	88.8
R-EWC - conv only	62.7	89.2	67.5	96.1	80.4	91.4	**84.7**	**93.1**	75.5	93.7
R-EWC - fc only	78.9	95.3	79.0	95.8	87.4	93.5	93.0	82.3	**94.3**	**88.0**
R-EWC - all	77.2	96.7	**91.7**	**91.2**	86.9	95.9	96.3	81.1	92.1	86.0
R-EWC - all no last	71.5	91.8	84.9	97.0	**91.6**	**94.5**	94.6	88.4	97.9	79.4

TABLE II: Comparison EWC / R-EWC for $T=2$.

	EWC [5] (T1 / T2)	R-EWC (T1 / T2)
MNIST	89.3 (85.8 / 92.8)	**93.1** (91.6 / 94.5)
CIFAR-100	37.5 (23.5 / 51.5)	**42.5** (30.2 / 54.7)
CUB-200 Birds	45.3 (42.3 / 48.6)	**48.4** (53.3 / 45.2)
Stanford-40 Actions	50.4 (44.3 / 58.4)	**52.5** (52.3 / 52.6)

2 a bit more difficult to learn. In conclusion, the improvement of our proposed method over EWC is that while maintaining similar task 2 performance, it allows for much less catastrophic forgetting on task 1. We have observed that during training the regularized part of the FIM is usually between 10^{-2} to 10^{-4}, which could explain why values around $\lambda = 100$ give a more balanced trade-off.

C. Comparison with EWC on two tasks

We further compare EWC and R-EWC on several larger datasets divided into two tasks – that is, in which all datasets are divided into 2 groups with an equal number of classes. The network is trained on task 1 as usual, and then both methods are applied for task 2. After the learning process is done, we evaluate the two tasks again. The accuracy for each task and average accuracy of two tasks are reported in Table II. Results show that our method clearly outperforms EWC for all datasets with an absolute gain on accuracy of R-EWC over EWC that varies from 2.1% to 5%. When comparing the accuracy on the first task only, R-EWC forgets significantly less in all cases while attaining similar accuracy on the second task.

D. Comparison with EWC on more tasks

We compare both EWC and R-EWC when having more tasks for datasets with larger images. We divide both CUB-200 Birds and Stanford-40 Actions datasets into four groups of an equal number of classes. We train a network on task 1 for both methods and proceed to iteratively learn the other tasks one at a time, while testing the performance at each stage. Results are shown in Figure 5, where we observe that the accuracy decreases with increasing number of tasks for both methods as expected.[4] However, R-EWC outperforms EWC consistently, by a margin that grows larger as more tasks are learned on Stanford-40 Actions dataset. Note that in lifelong learning settings it becomes more difficult to balance

[4]Fig. 5a has been corrected from the original version as it was duplicated.

(a) CUB-200 Birds
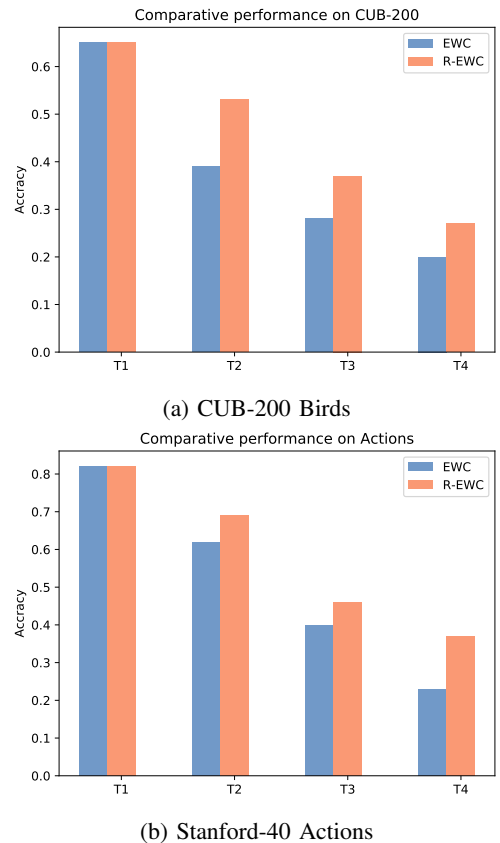

(b) Stanford-40 Actions

Fig. 5: Comparison with EWC when $T = 4$ on CUB-200 Birds and Stanford-40 Actions datasets.

performance of new tasks as the number of previous learned tasks increases. Results for all previous tasks after training the T-th task on Stanford-40 Actions are given in Table III. For each single previous task, our method manages to avoid forgetting better than EWC.

E. Comparison with the state-of-the-art

We compare our method (R-EWC) with fine-tuning (FT), Elastic Weight Consolidation (EWC) [5], Learning without Forgetting (LwF) [8] and Expert Gate (EG) [9]. We exclude methods which use samples of previous tasks during training of new tasks. We split the CIFAR-100 dataset [23] into 4 groups of classes, where each group corresponds to a task. For

TABLE III: Comparison EWC / R-EWC for $T = 4$ on Stanford Actions.

Current Task	Accuracy				
	T1	T2	T3	T4	Average
T1	81.5 / 81.5	-	-	-	81.5 / 81.5
T2	49.5 / **55.4**	75.8 / **81.5**	-	-	62.0 / **69.0**
T3	6.1 / **18.8**	45.1 / **48.7**	69.9 / **72.1**	-	40.3 / **47.2**
T4	0.0 / **12.0**	7.0 / **31.3**	44.5 / **56.9**	46.8 / **50.9**	23.0 / **37.2**

Fig. 6: Comparison with the state-of-the-art on CIFAR-100.

EG, the base network is trained on the 4 tasks independently, and for each of them we learn an auto-encoder with dimensions 4096, 1024 and 100, respectively. For FT, each task is initialized with the weights of the previous one. In addition, an UpperBound is shown by learning the newer tasks with all the images for all previous tasks available.

Results are shown in Figure 6, where we clearly see our method outperforms all others. FT usually performs worse compared to other baselines, since it tends to forget the previous tasks completely and is optimal only for the last task. EG usually has higher accuracy when the tasks are easy to distinguish (on CUB-200 and Stanford Actions, for example), however it is better than FT but worse than other baselines in this setting since the groups are randomly sampled from the CIFAR-100 dataset. EWC performs worse than LwF, however our method gains about 5% over EWC and achieves better performance than LwF. While our method still performs worse than the UpperBound, this baseline requires all data at all training times and can not be updated for new tasks.

VI. CONCLUSIONS

EWC helps to prevent forgetting but is very sensitive to the diagonal approximation of the FIM used in practice (due to the large size of the full FIM). We show that this approximation discards important information for preventing forgetting and propose a reparametrization of the layers that results in more compact and more diagonal FIM. This reparametrization is based on rotating the FIM in the parameter space to align it with directions that are less prone to forgetting. Since direct rotation is not possible due to the feedforward structure of the network, we devise an indirect method that approximates this rotation by rotating intermediate features, and that can be easily implemented as additional convolutional and fully connected layers. However, the weights in these layers are fixed, so they do not increase the number of parameters. Our experiments with several tasks and settings show that EWC in this rotated space (R-EWC) consistently improves the performance compared to EWC in the original space, obtaining results that are comparable or better than other state-of-the-art algorithms using weight consolidation without exemplars.

REFERENCES

[1] D. Silver and R. Mercer, "The task rehearsal method of life-long learning: Overcoming impoverished data," *Advances in Artificial Intelligence*, pp. 90–101, 2002.

[2] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *Psychology of learning and motivation*, vol. 24, pp. 109–165, 1989.

[3] S.-A. Rebuffi, A. Kolesnikov, and C. H. Lampert, "iCaRL: Incremental classifier and representation learning," in *CVPR*, 2017.

[4] D. Lopez-Paz and M. A. Ranzato, "Gradient episodic memory for continual learning," in *NIPS*, 2017, pp. 6470–6479.

[5] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, pp. 3521–3526, 2017.

[6] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang, "Overcoming catastrophic forgetting by incremental moment matching," in *NIPS*, 2017, pp. 4655–4665.

[7] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *ICML*, 2017, pp. 3987–3995.

[8] Z. Li and D. Hoiem, "Learning without forgetting," in *ECCV*, 2016, pp. 614–629.

[9] R. Aljundi, P. Chakravarty, and T. Tuytelaars, "Expert gate: Lifelong learning with a network of experts," in *CVPR*, 2017.

[10] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Advances in Neural Information Processing Systems*, 2017, pp. 2994–3003.

[11] A. Rannen, R. Aljundi, and M. B. B. T. Tuytelaars, "Encoder based lifelong learning," in *CVPR*, 2017, pp. 1320–1328.

[12] J. Serrà, D. Surís, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," *arXiv preprint arXiv:1801.01423*, 2018.

[13] S.-I. Amari, "Natural gradient works efficiently in learning," *Neural computation*, vol. 10, no. 2, pp. 251–276, 1998.

[14] R. Pascanu and Y. Bengio, "Revisiting natural gradient for deep networks," *arXiv preprint arXiv:1301.3584*, 2013.

[15] G. Desjardins, K. Simonyan, R. Pascanu *et al.*, "Natural neural networks," in *NIPS*, 2015, pp. 2071–2079.

[16] J. Lafond, N. Vasilache, and L. Bottou, "Diagonal rescaling for neural networks," *arXiv preprint arXiv:1705.09319*, 2017.

[17] R. Grosse and J. Martens, "A kronecker-factored approximate fisher matrix for convolution layers," in *ICML*, 2016, pp. 573–582.

[18] F. Huszár, "Note on the quadratic penalties in elastic weight consolidation," *Proceedings of the National Academy of Sciences*, vol. 115, no. 11, pp. E2496–E2497, 2018.

[19] R. K. Srivastava, J. Masci, S. Kazerounian, F. Gomez, and J. Schmidhuber, "Compete to compute," in *NIPS*, 2013, pp. 2310–2318.

[20] J. Li, "Restructuring of deep neural network acoustic models with singular value decomposition," in *Interspeech*, January 2013.

[21] M. Masana, J. van de Weijer, L. Herranz, A. D. Bagdanov, and J. M. Alvarez, "Domain-adaptive deep network compression," in *International conference on Computer Vision (ICCV)*, 2017.

[22] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[23] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[24] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," no. CNS-TR-2011-001, 2011.

[25] B. Yao, X. Jiang, A. Khosla, A. L. Lin, L. Guibas, and L. Fei-Fei, "Human action recognition by learning bases of action attributes and parts," in *ICCV*, 2011, pp. 1331–1338.

[26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[28] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," *arXiv preprint arXiv:1801.10112*, 2018.