Simplified computation and interpretation of Fisher matrices in incremental learning with deep neural networks

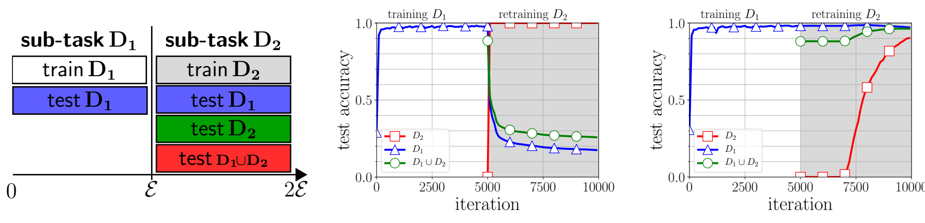
Alexander Gepperth and Florian Wiech

University of Applied Sciences Fulda, 36037 Fulda, Germany

Abstract. Import recent advances in the domain of incremental or continual learning with DNNs, such as Elastic Weight Consolidation (EWC) or Incremental Moment Matching (IMM) rely on a quantity termed the *Fisher information matrix* (FIM). While the results obtained in this way are very promising, the use of the FIM relies on the assumptions that **a)** the FIM can be approximated by its diagonal, and **b)** that FIM diagonal entries are related to the variance of a DNN parameter in the context of Bayesian neural networks. In addition, the FIM is notoriously difficult to compute in automatic differentiation (AD) systems frameworks like TensorFlow, and existing implementations require an excessive use of memory due to this problem. We present the Matrix of SQuares (MaSQ), computed similarly as the FIM, but whose use in EWC-like algorithms follows directly from the calculus of derivatives and requires no additional assumptions. Additionally, MaSQ computation in AD frameworks is much simpler and more memory-efficient FIM computation. When using MaSQ together with EWC we show superior or equal performance to FIM/EWC on a variety of benchmark tasks.

1 Introduction

This article describes a study in the context of incremental or continual learning with deep neural networks (DNNs). Essentially, this means that a DNN is not trained once, on a single task D, but successively on two or more sub-tasks $D_1, \dots, D_n$, one after another. Learning tasks of this type, which we term Sequential Learning Tasks (SLTs) (see Fig. 1a), are potentially very common in real-world applications. They occur wherever DNNs need to update their capabilities on-site and over time: gesture recognition, network traffic analysis, or face and object recognition in mobile robots. In such scenarios, neural networks have long been known to suffer from a problem termed "catastrophic forgetting"(CF) (e.g., [4]) which denotes the abrupt and near-complete loss of knowledge from previous sub-tasks $D_1, \dots, D_{k-1}$ after only a few training iterations on the current sub-task D_k (see Fig. 1b compared to Fig. 1c). In this article, we focus on SLTs from the visual domain with two sub-tasks each, as DNNs show pronounced CF behavior even when only two sub-tasks are involved.



(a) Training scheme (b) with CF (c) without CF

Fig. 1: Schema of incremental training experiments conducted in this article (a) and representative outcomes with (b) and without CF (c). The sequential learning tasks used in this study only have two sub-tasks: D_1 and D_2. During training (white background) and re-training (gray background), test accuracy is measured on D_1 (blue, △), D_2 (red, ○) and $D_1 \cup D_2$ (green, □). The blue curve allows to determine the presence of CF by simple visual inspection: if there is significant degradation w.r.t. the red curve, then CF has occurred.

1.1 Related work

The field of incremental learning is large, e.g., [16] and [6]. Recent systematic comparisons between different DNN approaches to avoid CF are performed in, e.g., [22,11] or [18]. Principal recent approaches to avoid CF include ensemble methods [21,2], dual-memory systems [23,10,19,5] and regularization approaches. Whereas [7] suggest Dropout for alleviating CF, the EWC method [13] proposes to add a term to the energy function that protects weights that are important for the previous sub-task(s). Importance is determined by a quantity that is claimed to approximate the Fisher information matrix of the DNN within a framework of Bayesian neural networks inspired by works on the natural gradient in DNNS [17]. A related approach is pursued by the Incremental Moment Matching technique (IMM) (see [15]), where weights from DNNs trained on current and past sub-tasks are “merged” using a similar approximation to the Fisher information matrix. Other regularization-oriented approaches are proposed in [1,24] and [12] which focus on enforcing sparsity of neural activities by lateral interactions within a layer.
Algorithms like EWC are in fact related to very old works on pruning neural network weights [8,20,9], where the same goal is pursued: to estimate how ”important” a weight is for the performance of the neural network by analyzing gradient information.

1.2 Motivation and goals of the article

We have been extensively analyzing [18] the performance of recently proposed algorithms for incremental learning like EWC or IMM, see Sec. 1.1. While doing so, we found that the computation of the FIM required for both methods is both computationally expensive, as well as conceptually questionable since the mathematical justification is at best unclear, and contains assumptions (diagonality of the FIM) that are neither proven nor empirically demonstrated. So the goal of the article is to propose a drop-in replacement for the FIM in regularization-based approaches to incremental learning like EWC or IMM that is both efficient to compute, and has a solid theoretical foundation which requires no unclear assumptions. The Matrix of SQuares (MaSQ) that we propose here has all of these properties, and we wish to empirically verify that incremental learning with EWC works just as well when using the MaSQ.

2 Methods

2.1 Dataset and construction of sequential learning tasks

We construct several sequential learning tasks (SLTs) from **MNIST** [14], a common benchmark for visual classification problems. It consists of 70.000 gray scale images of handwritten digits (0-9) of size 28x28, containing 55.000 training and 10.000 test samples that are approximately equally distributed over 10 classes. While MNIST may be considered too simple as an outright classification problem, we recently showed [18] that virtually all approaches to incremental or continual learning fail on simple two-task SLTs constructed from MNIST already, so MNIST-derived SLTs definitely do constitute adequate benchmarks here. We construct three types of two-task SLTs (defined by sub-tasks D_1 and D_2) from MNIST, which we term DP10-10 ("permuted"), D9-1 ("disjoint 9-1") and D5-5 ("disjoint 5-5"). The constructions given below apply equally to training, test and validation sets contained in MNIST.

Permuted SLT (DP10-10) This SLT is created by defining sub-task D_1 as the original MNIST benchmark containing all 10 classes, and adding sub-task D_2 as a copy of D_1 where copied samples all have their pixels spatially permuted in the same fashion. This is a benchmark that is widely used in studies on incremental learning, which we include for reference. As we could show[18], caution is required when using this benchmark as it seems to intrinsically facilitate incremental learning, probably because the patterns in D_1 and D_2 have very little overlap.

Disjoint SLTs (D9-1 and D5-5) These SLTs are created by defining D_1 as all samples from zero to eight (zero to four for D5-5) classes from MNIST, whereas D_2 is defined by the remaining classes (one for D9-1, five for D5-5).

2.2 DNN models and hyper-parameters

We employ simple fully-connected DNNs with EWC regularization, consisting of $L \in \{2, 3\}$ layers, all having an identical size of $S \in \{200, 400, 800\}$, and using the standard ReLU transfer function. We distinguish two learning rate parameters $\epsilon_1 = 0.001$ and $\epsilon_2 \in \{0.001, 0.0001, 0.00001, 1e-06\}$ for the two subtasks. Presence and effect of EWC regularization is governed by the balancing parameter $\lambda \in \{0, \frac{0.1}{\epsilon_1}, \frac{1}{\epsilon_1}, \frac{10}{\epsilon_1}\}$. Since we are dealing with classification problems, cross-entropy is used as a loss function $\mathcal{L}^{\mathrm{CE}}$. Mini-batch size in stochastic gradient descent (SGD) optimization is always set to $B = 100$.

2.3 Elastic Weight Consolidation with FIM and MaSQ

In [13,15], the EWC loss function for training on sub-task D_2 is given as

$$\mathcal{L} = \frac{\lambda}{2}\mathcal{L}^{\mathrm{CE}} + \sum_k F_k(\theta_k - \theta_k^{D_1})^2 \tag{1}$$

where the complete parameter set of a DNN (weight and biases) is denoted by $\boldsymbol{\theta}$, F_k describes the diagonal entries of the FIM (or MaSQ entries, see below), and $\boldsymbol{\theta}^{D_1}$ represents the complete set of stored parameters (again weights and biases) after having trained the DNN on sub-task D_1. In order to best describe our implementation of EWC (with FIM or MaSQ), we change the notation from the abstract parameter vector $\boldsymbol{\theta}$ of the DNN (as used in [13]) in favor of explicitly denoting the DNN weight matrices W^i and the bias vectors $\boldsymbol{b}^i, i \in \{0, \dots, L-1\}$. The EWC loss function used for re-training on D_2 then reads, in this notation:

$$\begin{aligned}\mathcal{L} = \mathcal{L}^{\mathrm{CE}} + \frac{\lambda}{2}\sum_{i=0}^{L-1}\sum_{kl} F_{kl}^{W^i}\left(W_{kl}^i - W_{kl}^{i,D_1}\right)^2 + \\ + \frac{\lambda}{2}\sum_{i=0}^{L-1}\sum_{k} F_{k}^{b^i}\left(b_{k}^i - b_{k}^{i,D_1}\right)^2\end{aligned} \tag{2}$$

where we introduce the "lagged variables" W^{i,D_1}, $\boldsymbol{b}^{i,D_1}$ as specified in [13], and the coefficient matrices F^{W^i} and coefficient vectors $\boldsymbol{F}^{b^i}$ that correspond to FIM diagonal elements or MaSQ entries for the different weight matrices and bias vectors, both computed after training on D_1 is completed.

We implement the EWC algorithm for a two-task SLT by the following strategy:

- train the DNN normally on D_1, using a balancing parameter of $\lambda = 0$
- copy weight matrices and bias vectors to the set of lagged variables
- perform a single pass through the training data (one epoch) without modifying weights or biases, for FIM or MaSQ computation (only the gradients are required)
- train the DNN on D_2, keeping the previous values of weights and biases, and using either FIM or MaSQ with a nonzero EWC balancing parameter λ
- test on D_2 and $D_1 \cup D_2$ during re-training on D_2

2.4 A critical discussion of FIM derivation and computation

The expression given in [13,15,17] for computing the FIM reads (again denoting the ensemble of DNN parameters as $\boldsymbol{\theta}$ as in [13]):

$$\mathcal{F}_{ij} \equiv \frac{1}{N}\sum_n \left(\frac{\partial \mathcal{L}}{\partial \theta_i}\frac{\partial \mathcal{L}}{\partial \theta_j}\Big|_{\boldsymbol{x}_n}\right), \tag{3}$$

where the expectation value is taken over all N training samples, indexed by n.

In [13,15], the FIM is assumed to be diagonal, so these authors use only the quantity

$$\boldsymbol{F} \equiv diag\left(\mathcal{F}_{ij}\right) \tag{4}$$

$$F_j \equiv \frac{1}{N}\sum_n \left(\frac{\partial \mathcal{L}}{\partial \theta_j}\Big|_{\boldsymbol{x}_n}\right)^2, \tag{5}$$

although this simplification is not proven, nor is it, given the generality of eqn.(3), very likely to hold in general. In our notation, the FIM (with diagonal assumption) is written as

$$\boldsymbol{F}^{W^i}_{jk} \equiv \frac{1}{N}\sum_n \left(\frac{\partial \mathcal{L}}{\partial W^i_{jk}}\Big|_{\boldsymbol{x}_n}\right)^2 \tag{6}$$

$$\boldsymbol{F}^{\boldsymbol{b}^i}_{j} \equiv \frac{1}{N}\sum_n \left(\frac{\partial \mathcal{L}}{\partial b^i_{j}}\Big|_{\boldsymbol{x}_n}\right)^2$$

We will verify FIM diagonality experimentally in Sec. 3.3.

A second point we like to raise is the utilization of the FIM diagonal in the EWC mechanism, given in its general form in eqn.(1), which is simply postulated in [13] and roughly justified as FIM diagonal entries F_k being equivalent to the certainty of parameter θ_k, and thus being a measure for its inverse variance in a Bayesian NN picture in [15]. The main justification of using FIM diagonal entries in [13,15] seems to be that the obtained results are very promising and give good results. We feel, however, that perhaps even better results could be obtained when using quantities in the EWC loss of eqn. (1) whose computation requires no diagonality assumptions, and whose use in eqn. (1) is justified by some rigorously provable mathematical principle. This is exactly what we propose with MaSQ, which will be detailed in the next section.

2.5 MaSQ computation and theoretical justification

As in [13], the DNN loss function $\mathcal{L}$ is considered to depend on a parameter vector $\boldsymbol{\theta}$, which, in reality, is a concatenation of all (flattened) weight matrices W^i and bias vectors $\boldsymbol{b}^i$ of the DNN. Since DNN loss functions are assumed to be differentiable almost everywhere at least once, we can apply the standard theory of differential calculus which states that a differentiable function such as $\mathcal{L}$ can be locally approximated by linear functions in all directions $\boldsymbol{\delta} \in \mathbb{R}^n$, see, e.g., [3]:

$$\forall \boldsymbol{\delta} \in \mathbb{R}^n : \mathcal{L}(\boldsymbol{\theta} + h\boldsymbol{\delta}) = \mathcal{L}(\boldsymbol{\theta}) + h\boldsymbol{J}\cdot\boldsymbol{\delta} + \frac{\eta(h\boldsymbol{\delta})}{h} \tag{7}$$

where we define a deviation parameter h, the gradient $\boldsymbol{J}$, $J_k = \frac{\partial \mathcal{L}}{\partial \theta_k}$ and a function $\eta(h\boldsymbol{\delta})$ that goes to zero faster than h as $h \to 0$. This formula indicates that $\mathcal{L}$ gets better and better approximated by the linear function $\mathcal{L}(\boldsymbol{\theta} + J \cdot \boldsymbol{\delta})$ as $h \to 0$. So, for sufficiently small h, the rate of change of $\mathcal{L}(\boldsymbol{\theta})$ as a reaction to small changes in a parameter θ_k are given by the gradient entry J_k. Identifying the parameters that contribute most to changes of $\mathcal{L}$ is then reduced to ranking the entries of $\boldsymbol{J}$ by their absolute value. Using the squared value for ranking is possible as well, since squaring is an operation that does not change the ranking.

Since gradients in DNN training are computed as training set (with N samples) expectation values over per-sample gradients J_{nk}, samples being denoted by $\boldsymbol{x}_n$,

$$J_k = \frac{1}{N}\sum_n J_{nk} = \frac{1}{N}\sum_n \left(\frac{\partial \mathcal{L}}{\partial \theta_k}\Big|_{\boldsymbol{x}_n} \right), \tag{8}$$

the squared gradients are then obtained as

$$F_k \equiv J_k^2 = \left(\frac{1}{N}\sum_n \frac{\partial \mathcal{L}}{\partial \theta_k}\Big|_{\boldsymbol{x}_n} \right)^2, \tag{9}$$

The squared entries of $\boldsymbol{J}$ can thus be directly used to punish changes to certain parameters more than changes to other parameters, since a higher value of J_k^2 will, by the definition of differentiable functions given in eqn. (7), depend quadratically on the modulus of the linear rate of change J_k of $\mathcal{L}$ upon small changes to the parameter θ_k. For this argument, it is immaterial whether the square or the modulus of J_k is used, although squares punish deviations for critical parameters more strongly. The squared entries J_k^2 thus form the Matrix of SQuares (MaSQ). Its entries F_k can be re-written in terms of the individual weight matrices and bias vectors as

$$\begin{aligned} \boldsymbol{F}^{W^i}_{jk} &\equiv \left(\frac{1}{N}\sum_n \frac{\partial \mathcal{L}}{\partial W^i_{jk}}\Big|_{\boldsymbol{x}_n} \right)^2 \\ \boldsymbol{F}^{\boldsymbol{b}^i}_{j} &\equiv \left(\frac{1}{N}\sum_n \frac{\partial \mathcal{L}}{\partial b^i_{j}}\Big|_{\boldsymbol{x}_n} \right)^2 \end{aligned} \tag{10}$$

in order to be inserted into the EWC loss function (2). When comparing eqns. (10) and (6), we note that MaSQ and FIM are actually computed in quite a similar fashion, as expectation values over loss gradients. However, FIM requires to square the gradients prior to taking the expectation value over training samples, whereas it is the other way round for MaSQ. Since, in practice, gradients are summed up over mini-batches and averaged after having traversed all training samples, FIM and MaSQ are equivalent for batch sizes of $B = 1$.

Memory consumption of FIM and MaSQ when using TensorFlow For $B > 1$ automatic differentiation frameworks like TensorFlow run into problems because they cannot compute per-sample gradients, which is required for FIM computation. So current reference implementations [1] use a workaround that consists of duplicating weight matrices and bias vectors B times, and then taking the gradient w.r.t each of the copies, which gives the per-sample gradients, although at the cost of a B-fold increase in memory consumption. MaSQ, in contrast, performs the squaring operation after having taken the average and thus does not require per-sample gradients to be computed.

3 Experiments

All DNN training is performed using stochastic gradient descent, and the Adam optimization strategy in particular. Training on D_1 or D_2 is always performed for 5000 iterations which, for MNIST, comes down to approximately 10 epochs. This value is empirically chosen, longer training times do not improve results. The code of our experiments is available on GitHub[2]. It is written in Python 3.6 using TensorFlow 1.12. The code is tested with GPU support, but will probably run without GPU support as well although much more slowly.
We perform our experiments in several steps:

- **Verification of incremental learning capacity of EWC/MaSQ** In Sec. 3.1, we test whether EWC learning on all three SLTs works with MaSQ. In order to make sure that results are generalizable (i.e., do not depend on a particular choice of hyper-parameters), we perform extensive hyper-parameter optimization w.r.t. DNN topology, and re-training learning rate.
- **Consistency check** In Sec. 3.2 we ensure that our EWC implementation is correct, by performing incremental learning experiments for the DNNs that performed best in the experiments of Sec. 3.1. This time, however, we work with a batch size of 1 for MaSQ computation, in which case, as outlined in Sec. 2.5, it corresponds exactly to the FIM as computed in [13].
- **Numerical comparison of FIM and MaSQ** In Sec. 3.4, we compare the numerical values computed for FIM and MaSQ on the same SLT to determine whether there are significant deviations. The reasoning for this is as follows: If there are no significant deviations between FIM and MaSQ, it is not surprising if there are no differences in EWC performance. However, if there are deviations but EWC works nevertheless with MaSQ, then we can conclude that MaSQ is a valid alternative to FIM.
- **Empirical check of FIM diagonality assumption** In Sec. 3.3, we check numerically whether the diagonal assumption made in [13,15] holds, at least approximately.

3.1 Verification of incremental learning capacity of EWC/MaSQ

For these experiments, we adhere to the full experimental paradigm outlined in Sec. 2.3, computing the MaSQ using a batch size of 1000. While training on

[1] https://github.com/stokesj/EWC
[2] www.github.com/EWC

b	SLT	L1	L2	L3	ϵ_2	accuracy in %	acc. in % for $\lambda = 0$
e	DP10-10	800	800	200	1e-05	96.94	x
s	D9-1	800	400	400	1e-04	96.93	x
t	D5-5	800	800	-1	1e-05	87.81	x
l	SLT	L1	L2	L3	ϵ_2	acc. in %	acc. in % for $\lambda = 0$
a	DP10-10	800	800	200	1e-04	96.84	89
s	D9-1	800	400	400	1e-05	96.27	30
t	D5-5	800	800	-1	1e-05	87.29	49

Table 1: Tabulated results of test accuracies for best DNN hyper-parameter settings, grouped by SLT. Evaluation is conducted using the "best" (above) or 'last" (below) strategies, see text for details. Results for $\lambda = 0$ are not given for the "best" strategy since it is inappropriate in this case (the test accuracy has no peak except at the beginning, which is meaningless). Entries of -1 for the size of layer 3 mean that this layer is absent.

all SLTs, hyper-parameter optimization is performed by exhaustively varying the parameters given in Sec. 2.2 within the given ranges, using classification accuracy as a selection criterion. We distinguish two possibilities for determining the quality of a particular training/retraining run: while always relying on the test accuracy on the whole dataset $D_1 \cup D_2$, one may consider the best or the last value of the re-training interval (assuming test accuracy is evaluated after each mini-batch iteration). While it is intuitive to use the best value, using the last value makes sense, too, since this quantity requires no extra effort to compute and is usually more robust to variations of the re-training interval. For completeness, the results given in Tab. 1 list both possibilities.

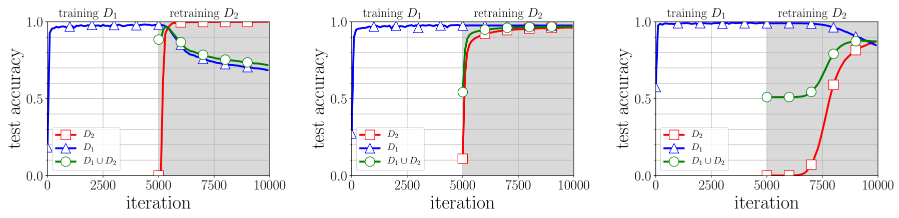


Fig. 2: Incremental learning performance of best DNNs (using EWC with MaSQ) resulting from hyper-parameter search EWC, applying the "best" criterion for evaluating an experiment. From left to right: D5-5, DP10-10, D9-1.

3.2 Consistency check

We select, for each SLT, the DNN that performed best in the hyper-parameter selection strategy of the previous section, using the "best" criterion. We then evaluate these three DNNs two times: one time with EWC turned off (balancing

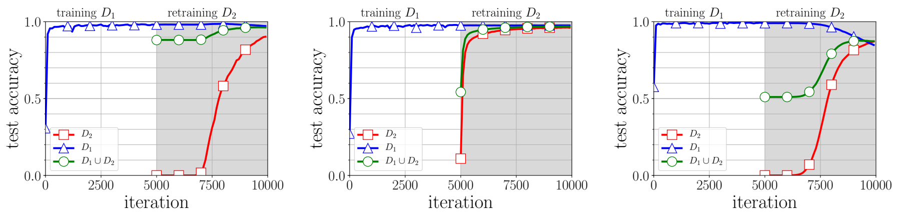


Fig. 3: Incremental learning performance of best DNNs (using EWC with MaSQ) resulting from hyper-parameter search EWC, applying the ”last” criterion for evaluating an experiment. From left to right: D5-5, DP10-10, D9-1.

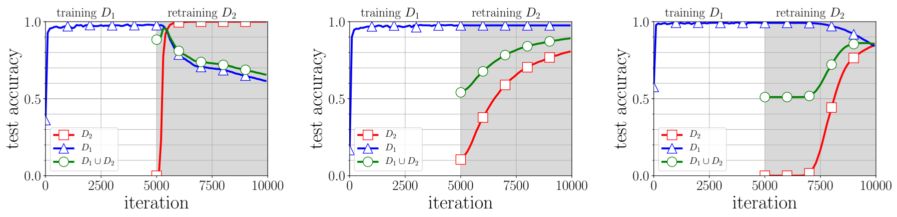


Fig. 4: Incremental learning performance of a DNN with fixed parameters with EWC, using the FIM instead of MaSQ. From left to right: D5-5, DP10-10, D9-1. To be compared to Figs. 2, 3 since the same hyper-parameters are used for each type of SLT.

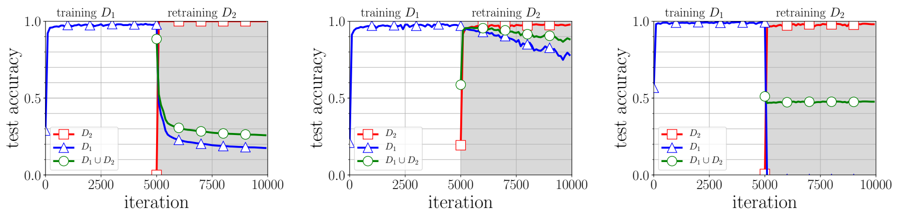


Fig. 5: Incremental learning performance of a DNN with fixed parameters without the EWC mechanism; i.e., setting $\lambda = 0$. From left to right: D5-5, DP10-10, D9-1. Strong forgetting can be observed for all SLTs. To be compared to Figs. 2, 3 since the same hyper-parameters are used for each type of SLT.

parameter $\lambda = 0$), and the second time with EWC turned on (using $\lambda = \frac{1}{\epsilon_2}$) but the batch size for MaSQ computation set to 1. In this case, the MaSQ and the FIM are identical, so we essentially perform EWC learning using the FIM. The results are given in Figs. 5, 4. We first observe in Fig. 5 that accuracy after re-training drops strongly for the D5-5 and D9-1 SLTs when turning off EWC. For DP10-10, forgetting is less strong, an effect already known from previous studies for this SLT [18]. In contrast, using the FIM together with EWC reduces forgetting for all SLTs and produces results that are very close to those obtained when using the MaSQ instead of FIM, see Figs. 2, 3. This shows that EWC with MaSQ produces similar results as EWC with FIM, so MaSQ can be considered a drop-in replacement.

3.3 Empirical check of FIM diagonality assumption

weight/bias→	W^3	$\boldsymbol{b}^1$	$\boldsymbol{b}^2$	$\boldsymbol{b}^3$
diag. sum	170.04	0.33	0.01	0.004
off-diag. sum	2023.41	1.36	0.04	0.004
diag. mean	0.002	0.0004	1.24e-06	4e-05
off-diag. mean	2.5e-07	1.7e-06	5.8e-08	4e-07

Table 2: Comparison of diagonal and off-diagonal entries of the FIM computed for SLT D5-5. For weight matrices W^1 and W^2, computation was too memory-consuming. We see that the average diagonal entry is, as a rule, several orders of magnitude larger than the off-diagonal entries. However, for the sums of diagonal and off-diagonal entries, this picture is reversed, a problem that grows worse with increasing number of weights in a DNN.

It is in principle rather simple to verify whether the FIM supports a diagonal assumption by evaluating eqn.(6) numerically (separately for each weight matrix and bias vector), although in practice memory limitations impose constraints: if a particular weight matrix W^i has dimension of, e.g., 100x100 weights, then the associated matrix $\mathcal{F}^{W^i_{kj}}$ would have 10000x10000 entries (approximately half a gigabyte at 32-bit floating point precision). In order to test FIM diagonality, we therefore use a very small, fixed DNN of dimensions 784-30-30-10 and train it on the SLT D5-5 for 5.000 iterations. Then we compute, for all weight matrices and bias vectors, the FIM $\mathcal{F}_{kl}$ defined in eqn.(3), with the parameter vector $\boldsymbol{\theta}$ being restricted to parameters from a particular weight matrix or bias vector.

The results for this very small DNN are given, for all SLTs, in Tab. 2. They show that, while individual diagonal entries of the full FIM are indeed much larger than off-diagonal entries, but that the sum of off-diagonal entries for exceeds the sum of diagonal entries. In an EWC-like mechanism including off-diagonal elements, these off-diagonal elements would therefore outweigh the diagonal elements, thus rendering the diagonal assumption questionable.

3.4 Numerical comparison of FIM and MaSQ

In order to perform a numerical comparison between the FIM and the MaSQ, we analyze both quantities for all weight matrices W^i and bias vectors $\boldsymbol{b}^i$ in the DNNs, again using the hyper-parameters of the DNNs that performed best in the hyper-parameter optimization of Sec. 3.1. The results for SLTs D5-5, D9-1 and DP10-10 are given in Figs. 6, 7, 8, respectively. We find a pronounced difference in maximal values of about a factor of 2, uniformly through all weight matrices, bias vectors and SLTs. This indicates that the FIM and the MaSQ are indeed substantially different quantities, and that the fact of EWC working with MaSQ is not because the MaSQ is equal, or proportional, to the FIM.

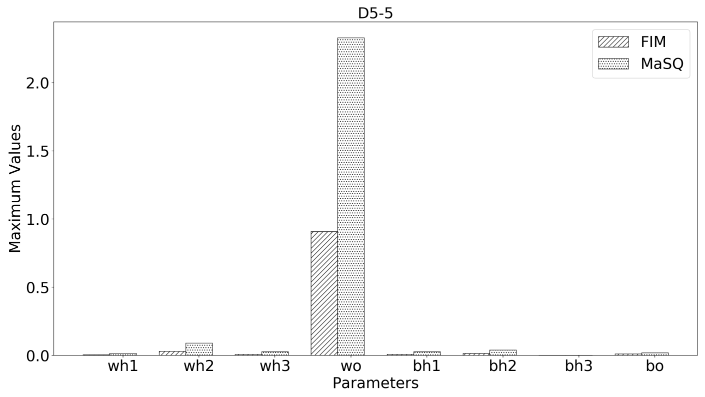


Fig. 6: Numerical comparison of maximal FIM and MaSQ values, given separately for all weight matrices (wh1:W^1,wh2:W^2,wh3:W^3,wo:W^4) and bias vectors (bh1:$\boldsymbol{b}^1$,bh2:$\boldsymbol{b}^2$,bh3:$\boldsymbol{b}^3$,bo:$\boldsymbol{b}^4$).

4 Discussion and principal conclusions

In this investigation, we introduce the Matrix of SQuares (MaSQ) as a drop-in replacement for the Fisher Information Matrix (FIM) in EWC-type incremental DNN learning algorithms. MaSQ is simple to compute and has a simple, mathematically well-founded interpretation.

MaSQ is effective in preventing catastrophic forgetting By the results of Sec. 3.1, we find that using MaSQ performs at least as good as FIM on all considered tasks, and that both effectively prevent catastrophic forgetting if correct parameter choices are made, which we do by an exhaustive search procedure.

MaSQ and FIM are different Results of Sec. 3.4 indicate that the MaSQ and the FIM are really numerically different, so similar performance cannot be

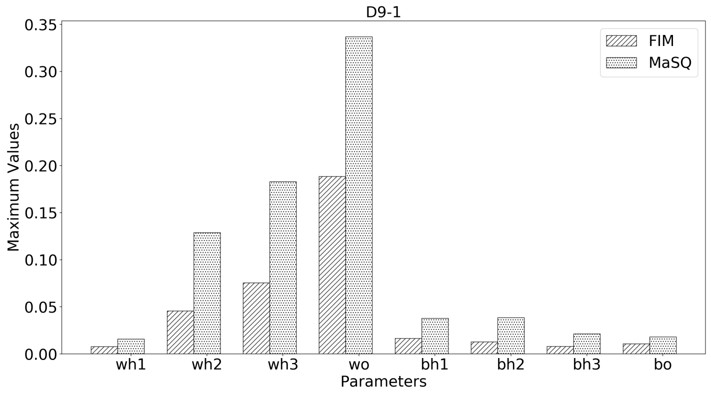


Fig. 7: Numerical comparison of maximal FIM and MaSQ values, given separately for all weight matrices (wh1:W^1, wh2:W^2, wh3:W^3, wo:W^4) and bias vectors (bh1:$\boldsymbol{b}^1$, bh2:$\boldsymbol{b}^2$, bh3:$\boldsymbol{b}^3$, bo:$\boldsymbol{b}^4$).

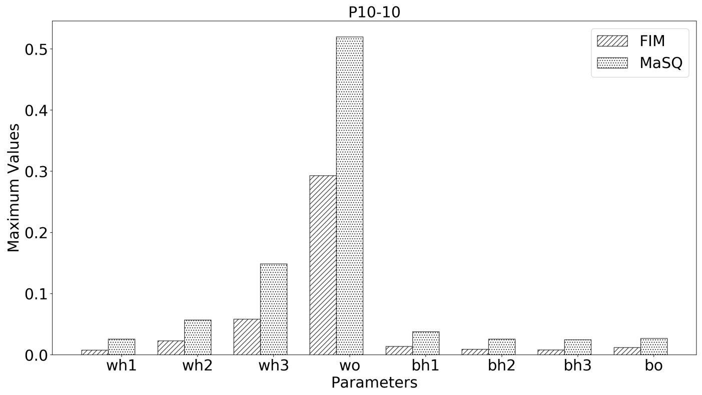


Fig. 8: Numerical comparison of maximal FIM and MaSQ values, given separately for all weight matrices (wh1,wh2,wh3,wo) and bias vectors (bh1,bh2,bh3,bo).

explained by numerical similarity, but rather by the fact that the use of the MaSQ is motivated by intuitive considerations about gradients of differentiable functions, in this case the DNN loss function $\mathcal{L}$.

The FIM diagonal assumption is problematic As we show for a simple setting in Sec. 3.3, the assumption that the FIM is diagonal is justified at first glance since diagonal elements are uniformly larger by at least an order of magnitude than off-diagonal elements. This is not really surprising since diagonal elements can only be positive due to the squaring, whereas off-diagonal elements have no such constraints and show lower average value simply because of this fact. We argue, however, that in an EWC-like mechanism using off-diagonal elements as well, these will have a much stronger impact since their number is far greater. Thus, neglecting the off-diagonal elements is not really justified when using the FIM for incremental EWC learning.

MaSQ is efficient and mathematically well-founded This is not really a problem when using the MaSQ, since it requires no diagonality assumption, and its use in EWC-like algorithms can be rigorously motivated from the calculus of derivatives. The fact that MaSQ computation is much more memory-efficient in frameworks like TensorFlow is an interesting by-product of our investigation.

Why FIM is problematic Even when leaving aside the issue of the diagonal assumption in FIM computation, one may ask whether there is really a big difference between eqns.(3) and (10): computing the expectation of squares or the square of the expectation does not really seem a noteworthy difference. However, when considering Jensen's inequality (which, as a particular case, states that for any convex function f and an integrable random variable X: $f(E(X)) \leq E(f(X))$), we can very easily construct cases where FIM would grossly overestimate the importance of a weights that is actually irrelevant. For example, let us consider the case where a weight's MaSQ value, i.e., $E[dL/dW])^2$, gives approximately 0, reflecting that W is unimportant w.r.t to the loss function. By Jensen's inequality, the corresponding FIM entry, that is, $E[(dL/dW)^2]$, can be arbitrarily large since it is the upper bound on the MaSQ value. This case occurs when there are both positive and negative contributions to the gradient that approximately cancel: with FIM, we would have assigned high importance to a weight that is actually useless.

Probabilistic interpretation of MaSQ What is often confusing is that the FIM is computed in terms of the loss function, yet is attributed a probabilistic meaning. To resolve this, is must be recalled that, in the probabilistic view of machine learning the loss function is defined as the log likelihood of the data under the model: minimizing the loss corresponds to maximizing this likelihood. From this probabilistic interpretation of the loss function, a probabilistic interpretation of the FIM and its various approximations may be motivated. MaSQ, on the other hand, relies solely on multi-variate calculus for its interpretation and treats the loss like any other multi-variate function, making it much clearer to see what MaSQ values actually mean: if they are high, the corresponding model parameter is important w.r.t. the loss, which may or may not have a probabilistic interpretation.

5 Future Work

A straightforward corollary of this article is that all algorithms that make use of the FIM should work just as well or better when using MaSQ. In particular, we will test this approach for the very promising IMM[15] algorithm that strongly relies on FIM. All things considered, MaSQ should actually show better performance for EWC and its variants, so another line of investigation will consist of analyzing and comparing MaSQ/EWC performance and comparing this to existing work in terms of accuracy, but also speed and memory consumption.

References

1. Aljundi, R., Rohrbach, M., Tuytelaars, T.: Selfless Sequential Learning (2018)
2. Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A.A., Pritzel, A., Wierstra, D.: PathNet: Evolution Channels Gradient Descent in Super Neural Networks (2017)
3. Forster, O.: Analysis 1, vol. 12. Springer (2004)
4. French, R.: Catastrophic forgetting in connectionist networks. Trends in Cognitive Sciences **3**(4), 128–135 (1999). https://doi.org/10.1016/S1364-6613(99)01294-2
5. Gepperth, A., Karaoguz, C.: A bio-inspired incremental learning architecture for applied perceptual problems. Cognitive Computation (2015), accepted
6. Gepperth, A., Hammer, B.: Incremental learning algorithms and applications. European Symposium on Artificial Neural Networks ({ESANN}) (April), 357–368 (2016)
7. Goodfellow, I.J., Mirza, M., Xiao, D., Courville, A., Bengio, Y.: An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks (2013). https://doi.org/10.1088/1751-8113/44/8/085201
8. Hassibi, B., Stork, D.G., Wolff, G.J.: Optimal brain surgeon and general network pruning. In: IEEE international conference on neural networks. pp. 293–299. IEEE (1993)
9. Karnin, E.D.: A simple procedure for pruning back-propagation trained neural networks. IEEE transactions on neural networks **1**(2), 239–242 (1990)
10. Kemker, R., Kanan, C.: FearNet: Brain-Inspired Model for Incremental Learning pp. 1–16 (2017)
11. Kemker, R., McClure, M., Abitino, A., Hayes, T., Kanan, C.: Measuring Catastrophic Forgetting in Neural Networks (2017). https://doi.org/10.1073/pnas.1611835114
12. Kim, H.E., Kim, S., Lee, J.: Keep and Learn: Continual Learning by Constraining the Latent Space for Knowledge Preservation in Neural Networks (2018)
13. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., Hadsell, R.: Overcoming catastrophic forgetting in neural networks (2016). https://doi.org/10.1073/pnas.1611835114, http://arxiv.org/abs/1612.00796
14. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-Based Learning Apllied to Document Recognition (1998)
15. Lee, S.W., Kim, J.H., Jun, J., Ha, J.W., Zhang, B.T.: Overcoming Catastrophic Forgetting by Incremental Moment Matching (Nips), 1–16 (2017)

16. Parisi, G.I., Kemker, R., Part, J.L., Kanan, C., Wermter, S.: Continual Lifelong Learning with Neural Networks: A Review pp. 1–29 (2018)
17. Pascanu, R., Bengio, Y.: Revisiting natural gradient for deep networks. arXiv preprint arXiv:1301.3584 (2013)
18. Pfülb, B., Gepperth, A.: A comprehensive, application-oriented study of catastrophic forgetting in dnns. In: International Conference on Learning Representations (ICLR) (2019), accepted
19. Rebuffi, S.a., Kolesnikov, A., Sperl, G., Lampert, C.H.: iCaRL : Incremental Classifier and Representation Learning pp. 2001–2010 (2017)
20. Reed, R.: Pruning algorithms-a survey. IEEE transactions on Neural Networks **4**(5), 740–747 (1993)
21. Ren, B., Wang, H., Li, J., Gao, H.: Life-long learning based on dynamic combination model. Applied Soft Computing Journal **56**, 398–404 (2017). https://doi.org/10.1016/j.asoc.2017.03.005
22. Serrà, J., Surís, D., Miron, M., Karatzoglou, A.: Overcoming catastrophic forgetting with hard attention to the task. arXiv preprint arXiv:1801.01423 (2018)
23. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual Learning with Deep Generative Replay (Nips) (2017)
24. Srivastava, R.K., Masci, J., Kazerounian, S., Gomez, F., Schmidhuber, J.: Compete to Compute. Nips pp. 2310–2318 (2013)