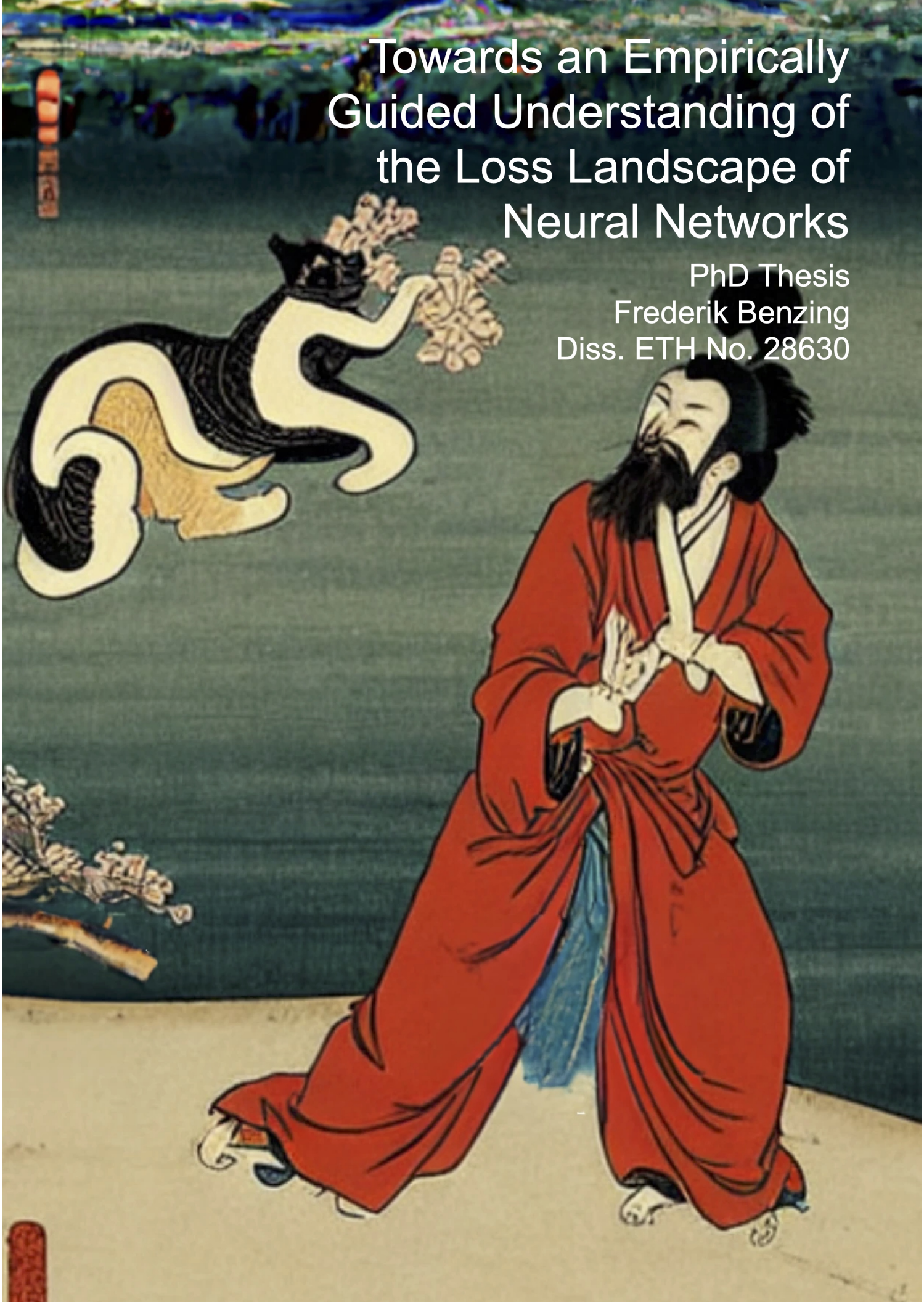


Towards an Empirically Guided Understanding of the Loss Landscape of Neural Networks

PhD Thesis
Frederik Benzing
Diss. ETH No. 28630

DISS. ETH NO. 28630

TOWARDS AN EMPIRICALLY GUIDED UNDERSTANDING OF THE LOSS LANDSCAPE OF NEURAL NETWORKS

A dissertation submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by

FREDERIK BENZING
MSc, ETH Zürich

born on 05 February 1994
citizen of Germany

accepted on the recommendation of

Prof. Dr. Angelika Steger, examiner
Prof. Dr. Laurence Aitchison, co-examiner
Dr. Razvan Pascanu, co-examiner

2022

Cover Design: "Confucius fighting a cat" created with Stable Diffusion and DALL·E 2.

DOI: 10.3929/ethz-b-000572885

ABSTRACT

One of the most important and ubiquitous building blocks of machine learning is gradient based optimization. While it has and continues to contribute to the vast majority of recent successes of deep neural networks, it comes both with some limitations and the potential for further improvements.

Catastrophic forgetting, which is the subject of the fist two parts of this thesis, is one such limitation. It refers to the observation that when gradient based learning algorithms are asked to learn different tasks sequentially, they overwrite knowledge from earlier tasks. In the machine learning community, several different ideas and formalisations of this problem are being investigated. One of the most difficult versions is a setting in which the use of data from earlier distributions is strictly forbidden. In this domain, an important line of work are so-called regularisation based algorithms. Our first contribution is to unify a large family of these algorithms by showing that they all rely on the same theoretical idea to limit catastrophic forgetting. This had not only been unknown, but we also show how this is an accidental feature of at least some of the algorithms. To demonstrate the practical impact of these insights, we also show how they can be used to make some algorithms more robust and performant across a variety of settings.

The second part of the thesis uses tools from the first part and tackles a similar problem, but does so from a different angle. Namely it focusses on the phenomenon of catastrophic forgetting – also known as the stability-plasticity dilemma – from the viewpoint of neuroscience. It proposes and analyses a simple synaptic learning rule, based on the stochasticty of synaptic signal transmission and shows how this learning rule can alleviate catastrophic forgetting in model neural network. Moreover, the learning rule's effects on energy-efficient information processing are investigated extending prior work which explores computational roles of the aforementioned and somewhat mysterious stochastic nature of synaptic signal transmission.

Finally, the third part of the thesis focuses on potential improvements of standard first-order gradient based optimizers. One of the most successful lines of work in this area are Kronecker-factored optimizers, whose influence has reached beyond optimization to areas like Bayesian machine learning, catastrophic forgetting or meta-learning. Kronecker-factored optimizers

are motivated and thought of as approximations of natural gradient descent, a well-known second-order optimization method. We will show that a host of empirical results contradict this view of KFAC as a second-order optimizer and propose an alternative, fundamentally different theoretical explanation for its effectiveness. This does not only give important new insights into one of the most powerful optimizers for neural networks, but can also be used to derive a more efficient optimizer.

ZUSAMMENFASSUNG

Einer der wichtigsten und allgegenwärtigen Bausteine des maschinellen Lernens ist die gradientenbasierte Optimierung. Obwohl sie weiterhin zu den meisten jüngsten Erfolgen von tiefen neuronalen Netzen beiträgt, weist sie auch einige Einschränkungen und das Potenzial für weitere Verbesserungen auf.

Eine solche Einschränkung ist das "katastrophale Vergessen", das Gegenstand der ersten beiden Teile dieser Arbeit ist. Es bezeichnet die Beobachtung, dass gradientenbasierte Lernalgorithmen, wenn sie verschiedene Aufgaben nacheinander lernen sollen, das Wissen aus früheren Aufgaben überschreiben. In der Forschung zu maschinellem Lernen werden mehrere verschiedene Ideen und Formalisierungen dieses Problems untersucht. Eine der schwierigsten Varianten ist ein Rahmen, in dem die Verwendung von Daten aus früheren Aufgaben streng verboten ist. In diesem Bereich sind die so genannten regularisierungsbasierten Algorithmen ein zentraler Bestandteil des aktuellen Forschungsstandes. Der erste Beitrag dieser Arbeit besteht darin, eine große Familie dieser regularierungsbasierten Algorithmen zu vereinigen, indem wir zeigen, dass sie alle auf der gleichen theoretischen Idee beruhen, um das katastrophale Vergessen zu begrenzen. Um praktische Anwendungen dieser Erkenntnisse zu demonstrieren, zeigen wir auch, wie sie genutzt werden können, um einige Algorithmen in einer Vielzahl von Situationen robuster und leistungsfähiger zu machen.

Der zweite Teil dieser Arbeit verwendet die Werkzeuge aus dem ersten Teil und geht ein ähnliches Problem an, allerdings aus einem anderen Blickwinkel. Er konzentriert sich nämlich auf das Phänomen des katastrophalen Vergessens – auch bekannt als das Stabilitäts-Plastizitäts-Dilemma – aus der Sicht der Neurowissenschaften. Es wird eine einfache synaptische Lernregel vorgeschlagen und analysiert, die auf der Stochastizität der synaptischen Signalübertragung beruht, und es wird gezeigt, wie diese Lernregel das katastrophale Vergessen in einem neuronalen Modellnetzwerk abmildern kann. Darüber hinaus werden die Auswirkungen der Lernregel auf die Energieeffizienz von Informationsverarbeitung untersucht und damit frühere Arbeiten erweitert, die die rechnerische Rolle der oben erwähnten und rätselhaften stochastischen Natur der synaptischen Signalübertragung untersuchen.

Schließlich konzentriert sich der dritte Teil der Arbeit auf mögliche Verbesserungen von weitverbreiteten gradientenbasierten Optimierungsalgorithmen erster Ordnung. Einer der erfolgreichsten Forschungszweige in diesem Bereich sind Kronecker-faktorisierte Optimierungsalgorithmen, deren Einfluss über die Optimierung hinaus in Bereiche wie Bayes'sches maschinelles Lernen, katastrophales Vergessen oder Meta-Lernen reicht. Kronecker-faktorisierte Optimierer sind als Annäherungen an das "natürliche Gradientenverfahren" motiviert, eine bekannte Optimierungsmethode zweiter Ordnung. Wir werden zeigen, dass eine Vielzahl empirischer Ergebnisse dieser Sichtweise von KFAC als Optimierer zweiter Ordnung widerspricht und schlagen eine alternative, grundlegend andere theoretische Erklärung für KFACs Effektivität vor. Dies gibt nicht nur wichtige neue Einblicke in einen der leistungsstärksten Optimierungsalgorithmen für neuronale Netze, sondern kann auch zur Herleitung eines effizienteren Optimierungsverfahrens verwendet werden.

ACKNOWLEDGEMENTS

First and foremost, I thank Angelika for being a role model both as a scientist and boss: As a scientist, I admire your curiosity, originality and your dedicated, inspiring teaching. I appreciate the freedom that you gave me to change topics and to pursue what I found interesting. As my boss, you have created an incredibly friendly, supportive, constructive and, of course, smart group of people. I always felt sure — and saw several times — that if needed you would be on our side and protect us and our well-being without hesitation. I find it hard to imagine a nicer team, spirit and environment to do research in.

I am also very thankful to Laurence, for sharing your time and thoughts with me countless times. I learned a lot from our motivating discussions, your insights, input as well as your critical and always helpful feedback. My favourite part of this thesis certainly wouldn't have seen the light of day without your input, and, also without this, working with you made my time more productive and enjoyable.

Last, but not least, I am incredibly thankful to all my colleagues. Again, I find it difficult to imagine a nicer and more fun environment to work in.

CONTENTS

1

INTRODUCTION

Many technological revolutions in the history of humanity can be traced backed to a single, deceptively simple, idea. Take the agricultural or industrial revolution as examples – the basic ideas underlying those are as simple as "plant food" and "transform heat energy into mechanical energy". For a more recent example, even the digital revolution was arguably sparked by one simple (albeit not trivial) idea, namely that to formalise and automate computation.

A new large wave of technological innovation that is about to happen is that of machine learning. Of course, it is hard to predict if its impact will be anywhere near that of the revolutions described above. Nevertheless, it has become difficult to imagine a future in which deep learning techniques will not become ubiquitous in one way or the other[1]. And even without knowing the future impact of deep learning, nothing stops us from wondering what the key ideas underlying its recent successes are. Of course, most researchers would not enjoy answering this questions, as it comes with the need to ignore and simplify all too many things. Nevertheless, if pressed to do the impossible, many answers would likely be along the lines of "Gradient-based learning in (large) neural networks on (large) datasets gives useful representations of the world".

This answer, like most short answers to difficult questions, is not only the ruthless simplification we asked for but also a bit of a cheat, as it includes at least two ideas that would deserve a great deal more of elaboration – that of neural networks and that of gradient-based learning. The reason for this simplification is of course to set the stage for the present thesis, which will be concerned with some properties and potential weaknesses of gradient based learning, undoubtedly one of the key ingredients of deep learning.

Before we move on to focussing on gradient based learning, a few things are worth making explicit: By calling ideas simple, we do not intend to diminish their importance or ingenuity. Neither do we want to give the impression that the initial idea is all you need – recognising its potential, getting its details right, developing, popularising and scaling it are all indispensable components that underlie all revolutions mentioned above.

1 To the extent to which they aren't already.

Without them, we would not have gotten from planting food to genetically modifying crops, from steam-powered train engines to space travel, or from Leibniz' calculating machine to a global server infrastructure that answers billions of search queries each day. Similarly, all these steps have contributed to the journey from plain perceptrons to capable vision-, language- or reinforcement learning models trained with large amount of data on specialised and ever more efficient hardware.

1.1 GRADIENT BASED LEARNING

The first step of learning in neural networks is defining a suitable objective function that the network has to optimize. Especially in self-supervised learning this has been an important area of research and key to learning useful representations across different modalities and also in multi-modal settings [1–3]; here we will skip over this important step and assume we are given a reasonable and useful objective function.

Once an objective is defined, the next step is to optimize it. From a theoretical perspective, this can easily seem like a hopeless endeavour: Due to the (indispensable) non-linearities of neural networks, their loss-landscape is non-convex and optimizing them is NP-hard [4]. Even worse, already for moderately sized networks any kind of global- / brute-force search in parameter space is infeasible, and will remain infeasible as long as computing devices obey the known laws of physics.

At this point theoreticians might shrug their shoulders and turn their back on neural networks. Luckily, practitioners have long known gradient-based learning as a simple heuristic for optimization [5]. Even though gradient-based learning is a simple answer to a complex problem, this time the idea really is almost as simple as it sounds: Compute the gradient of the objective with respect to the parameters and update the paremter in gradient direction. Empirically, only two adaptions to vanilla gradient descent seem to be necessary to obtain a very strong, competitive optimizer: Computing gradients on mini-batches, i.e. performing stochastic gradient descent[2] and adding momentum [7]. For example, evidence from [8] suggests that SGD with Momentum comes within a very close margin of more sophisticated optimizers in both vision and language tasks.

While SGD (or a variant thereof) is a the key component of almost every success of deep learning, it does come with a few potential shortcomings. The next two sections will each describe one such shortcoming, a line of

2 This may not be strictly necessary, but almost certainly makes generalisation easier, see e.g. [6]

work trying to overcome that shortcoming and how the work presented in this thesis contributes to these lines of work.

1.2 SGD FORGETS (CATASTROPHICALLY)

SGD is a strictly local search algorithm and as hinted at previously, any global understanding or evaluation of the loss landscape remains computationally infeasible. One consequence of this is that even if we have successfully optimized a loss function, our knowledge about the loss landscape remains very limited. For example, if we are given a new set of parameters, we typically cannot infer how this set of parameters will perform unless we explicitly evaluate the loss again.

To see why this can be problematic, imagine a situation in which we have a network which has already learned some task (i.e. it achieves low loss on some distribution) and which should learn an additional task. To learn the new task and integrate new knowledge, the network's parameters have to be updated, but as soon as we walk away from our prior solution, we have no idea if the new parameter values will retain a low loss on the previous objective. Thus, if we are not allowed to evaluate the previous loss function again, learning the new task is prone to gradually overwrite prior knowledge.

This phenomenon is often referred to as "Catastrophic Forgetting" or "Continual Learning" and has long been known, e.g. [9]. Recently, due to works like [10–12] the problem has gained more attention and inspired a by now considerable body of literature. Typically, a network is trained on different data distributions sequentially, and when learning new datasets the access to old ones is limited. Then after training, the network is tested on data from both recent and old distributions. There is a number of different formalisations of this setting and correspondingly different experimental test beds for algorithms attempting to overcome this problem. One particularly important difference between settings and algorithms is how strictly they limit access to data from prior distributions – the strictest version of the problem, that arguably remains an open, difficult and important research question, forbids storing examples from prior distributions altogether.

An important line of work in this difficult domain pioneered by [11] proposes using a local approximation of each distribution's loss as a proxy for the actual loss. Then, when a new task arrives, the network is jointly trained on the new task's loss and the approximation of prior losses.

After [11] a large number of variations of this idea have been published [e.g. [13–22]] and – at first sight – it appears that each of these variations provides new insights into the loss landscape of neural networks, as each variation suggest a different view on which local properties of the loss are important and help to limit forgetting. In Chapter 3 we will take a closer look at some popular algorithms from this domain and argue both theoretically and empirically that these algorithms [13, 14] (and consequently their follow up work) use the same idea, first proposed in [11], and rely on a simple second-order taylor approximation of the loss. Equivalently, using the bayesian framing from [11], they all rely on a Laplace approximation of posterior likelihoods.

Theoretically, this insight has the benefit of giving a unified, simplified view of a large family of algorithms and it brings more clarity to the question which local properties of the loss landscape matter. From a more applied perspective, we also demonstrate empirically in Chapter 3 how these contributions can be used to make one of the algorithms more robust and performant across different settings.

1.2.1 *Catastrophic Forgetting and Biological Neural Networks*

Above we have focussed on artificial neural networks in the context of deep learning and described the tension between adapting network parameters to integrate new knowledge and keeping parameters fixed to preserve old knowledge. The very same tension is present in biological neural networks, and in neuroscience the corresponding problem is known as the stability-plasticity dilemma.

It is clear from observation that biological neural networks (at least in many situations) can handle this trade off rather well: For example, once a human learns to ride a bike, they will hardly unlearn it, even if they do not cycle for years and integrate large amounts of new knowledge into their motor-cortex by acquiring new motions patterns. While the fact that biological neural networks can trade of stability and plasticity effectively is hardly disputable, the question how they achieve this feat is an open and important one in the neuroscience community.

In Chapter 4, we will turn towards the context of biological neural networks and investigate a simple learning rule that could alleviate the tension between stability and plasticity in model neural networks. While several experiments and additional contributions in Chapter 4 are explicitly

tailored towards neuroscience, we will also use tools from machine learning and rely on insights from Chapter 3 to analyse the proposed learning rule.

More context on how this work is situated with respect to other studies in neuroscience and which additional contributions will be made are deferred to Chapter 4.

1.3 SGD MAY BE INEFFICIENT IN ILL-CONDITIONED LOSS LANDSCAPES

A key component of scaling and improving neural networks is increasing their depth. But with great depth come optimization difficulties, as the loss landscape of neural networks becomes "less smooth" and more poorly conditioned. Overcoming the trainability issues of deep neural network has inspired several papers and techniques that have proven hugely influential for the entire field of deep learning [23–28]. Roughly speaking, three key ideas have emerged to make deep neural networks trainable with SGD: (1) Careful weight-initialisation schemes [24, 25] (2) Skip-connections [23, 28] (3) Normalisation Layers [26, 27].[3][4]

Careful initialisation is generally motivated by preserving signal propagation across the entire depth of the network and by avoiding exploding or vanishing gradients [24, 30, 31]. Skip connections have similar effects, and additionally, like normalisation, they are thought to lead to smoother, better conditioned loss landscapes, see e.g. [32, 33].

Going back to the viewpoint presented at the beginning that machine learning relies on two key ideas – neural networks and gradient-based learning – it becomes clear that the solution to training deep neural networks described above focusses purely on tuning the first idea: All three components – weight initialisation, skip-connections and normalisation[5] – change the structure of the neural network. The second idea, gradient-based learning, remains untouched and untuned.

This view directly suggests a different approach to training deep neural networks: We could try to leave the neural network part untouched and

3 In this context it is also interesting that a key component of Batch-Normalisation is that it effectively changes the initialisation scheme in architectures with skip connections, as clearly explained in [29].

4 This list is not complete and omits for example pretraining early layers, but it represents current best-practices, which seem to be sufficient.

5 Whether normalisation layers are an architectural- or optimization-technique is somewhat debatable. Normalisation can usually be viewed as a change in parametrisation, which is often equivalent as a change in optimizer.

instead improve upon SGD by using a different, better gradient-based optimization algorithm, that can handle the complexities of deep loss landscapes.

This line of work has indeed been pursued and recent results [34, 35] suggest that using better optimizers, together with careful weight initialisation, may be equally effective and fast at training deep networks as the alternative described at the beginning of this section.

1.3.1 *Why does improving SGD matter?*

If we can already train deep networks with SGD by modifying the neural network architecture, why should we care about finding an alternative way to do so by improving optimization? Below, we will exemplify three answers to this question whose motivations range from that of a fundamental researcher to that of a large company employing models at scale.

(1) Given the immense importance of training deep neural networks, it almost has to be an intrinsic interest of a research community to find a few different ways to do so.

(2) Finding a new solution has the potential of solving some of the shortcomings of the old one – for example, batch norm, a key architectural component in deep vision networks, is sometimes argued to hamper transfer learning [36], and when we find a new way to train deep networks, we might be able to solve this problem. Along a similar line of thought, it might even be that we are not aware of the present solution's limitations before comparing it to the new solution.

(3) An optimizer that works well in complex loss landscapes, may very well lead to training speed-ups in simpler loss landscapes. Given that essentially every deep learning paper uses gradient-based optimization, speeding up training meaningfully would allow considerably faster, and thus more experimentation, and would likely have a huge impact on the entire field. Moreover, at the enormous scale at which current models operate, training a single model can take months, cost tens of millions of dollars and emit hundreds of tons of CO_2 (see e.g. [37]). Some recent, informally published results [38] indicate that employing better optimizers at a large scale could speed up training significantly, and thus could lead to notable financial and environmental savings.

1.3.2 *How can SGD be improved?*

As emphasised previously, obtaining a global understanding of the loss landscape of neural networks is prohibitively difficult. From this viewpoint, it is then clear that one way to improve optimizers is gaining a better understanding of local properties of the loss landscape. Based on a better local understanding, we might be able to find update directions which allow for faster progress during optimization.

One well-known way to view SGD is that it approximates the loss function by its first-order Taylor expansion and then optimizes this first-order approximation subject to staying within a certain "trust region" around the previous parameter point, in which the first-order approximation is reliable. The size of this trust region then directly corresponds to the step-size of SGD.

From this perspective, a well known and natural way to improve upon SGD, which dates back already to Gauss, who adapted an idea of Newton, is to use a higher-order Taylor expansion of the loss function, which will then be optimized. Unfortunately, already for the second-order Taylor expansion this approach quickly faces some serious problems: Storing the second-derivative requires n^2 numbers, where n is the number of parameters of the neural network. Moreover, to minimize the second-order Taylor expansion, we need to invert the second-derivative, which naively takes time n^3. Neither the space nor the time requirement of this algorithm is in any way feasible for modern neural networks, which typically have millions to hundreds of billions of parameters[6].

Thus, to use second-order optimization it is necessary to either not compute and invert the hessian explicitly (e.g. [39, 40]) or to approximate it further (e.g. [41, 42]). The second approach of approximating second-derivatives further has lead to a particularly successful line of work [41, 42], namely that of Kronecker-factor optimizers. This line of work is also what underlies the success of training deep networks without the usual tricks of normalisation and skip-connections mentioned above [34, 35].

In Chapter 5, we will take a close look at those optimizers. We will describe the surprising, paradoxical discovery that the approximations made by Kronecker-factored optimizers make them perform considerably better than their exact second-order counterpart, which does not use these approximations. This strongly suggests that Kronecker-factored optimizers do not actually exploit the second-order Taylor expansion to improve optimization,

6 at the time of writing

but – due to their approximations – pick up on a different, meaningful local property of the loss landscape. In Chapter 5 we will also develop a theoretical hypothesis for what this local property could be and present empirical evidence that this hypothesis explains large parts of the behaviour of Kronecker-factored optimizers.

From a theoretical perspective, these results are interesting as they suggest that the standard second-order Taylor approximation, despite being the theoretically most natural approach, is not necessarily the most useful local model of neural networks' loss landscapes. This is also a crucial insight to inform future work, as it suggests strongly that developing better approximations of the second-order Taylor approximation will not be useful, and that instead, different properties of the loss landscape have to be explored. From a more practical perspective, we will also show how our insights lead to a computationally cheaper and often more effective optimizer and note that similar optimizers had already been explored in the literature [42, 43].

1.3.3 *Link to Catastrophic Forgetting*

At this point, it is worth making explicit that Chapters 3 and 5 share a key idea. In both settings, the idea is to use a good, local approximation of the loss landscape – once to avoid forgetting and once to improve optimization. Entertainingly, Chapter 3 will argue that a set of algorithms relies on using the second-order Taylor approximation, despite not explicitly being designed to do so, while Chapter 5 will argue that a set of algorithms does not use the second-order Taylor approximation, despite being explicitly designed to do so.

1.4 METHODOLOGY

The exposition above should explain why the title of this thesis features the "loss landscape of neural networks" and in how far understanding it motivates the contributions of this thesis. In addition to being linked in terms of motivation and topic, Chapters 3, 5 and also Chapter 4 share a large part of their methodology. The contributions of these chapters are at the intersection of theoretical and practical insights and a key feature will be developing better theoretical explanation for empirically observed behaviours of neural network algorithms. Therefore, we will put significant emphasis on designing experiments to explicitly test theoretical hypothesis.

This is what explains why the words "empirically guided" also feature in the title of this thesis and is a somewhat under-represented approach[7] in a benchmark driven community. We will not ignore the communities thirst for benchmark results and explain and exemplify how our theoretical insights can be used to improve the algorithms investigated. At the same time, we also argue that understanding algorithms has intrinsic value as part of the scientific endeavour and should receive more attention in the deep learning community.

1.5 OVERVIEW OF REMAINING CHAPTERS

Above, we have introduced the topics of this thesis from one particular angle and explained how both continual learning and optimization are closely linked to obtaining a better understanding of local properties of the loss landscape of neural networks. As a consequence, we have at least partially neglected to present each contribution fully within the scope of its related work. Therefore, Chapters 3, 4 and 5 will have a separate introduction, related work section and discussion section to fill this gap.

Before we move on to our contributions to Continual learning within machine learning (Chapter 3), continual learning within neuroscience (Chapter 4) and neural network optimization (Chapter 5), we will present some background on the Fisher Information (Chapter 2) as a key tool for local loss approximations that is used in all following chapters.

7 but of course not unique

2

BACKGROUND

Some parts of this chapter are taken and adapted from Benzing [44]. Note that none of the insights presented here are new and they should be attributed to the sources cited in the text.

All following chapters will in one way or another be concerned with approximations of the second-order Taylor expansion of the loss of neural networks, and here we will provide some brief background to make precise what these approximations are, why they appear and how they are motivated.

We first need some notation and definitions: We will use $\mathbf{w}$ to denote the parameters of the neural network. For a pair of input and label (X, y)[1] we will denote by $\ell(y \mid X, \mathbf{w})$ the loss that the network has on this pair. In our case, the loss will always be the negative log-likelihood.

We will use the shorthand $\mathbf{g}(X, y)$ for the "gradient", i.e. $\mathbf{g}(X, y) = \frac{d\ell(y|X,\mathbf{w})}{d\mathbf{w}}$ and we will write $\mathbf{H} = \frac{d^2\ell(y|X,\mathbf{w})}{d\mathbf{w}^2}$ for the Hessian.

Then, the second-order Taylor expansion of the loss is of course given by

$$\ell(y \mid X, \mathbf{w}_0 + \mathbf{w}) \approx \ell(y \mid X, \mathbf{w}_0) + \mathbf{g}(X, y)^T \mathbf{w} + \frac{1}{2} \mathbf{w}^T \mathbf{H} \mathbf{w} \tag{2.1}$$

If we want to use this approximation for continual learning or optimization there is an immediate problem: The Hessian might have negative eigenvalues and this is undesirable both in the context of continual learning and optimization. For optimization, negative eigenvalues imply that there is no minimizer of the second-order expansion and that the optimizer would want to take infinitely large update steps in the direction of eigenvectors with negative eigenvalues. In practice, this would lead to unstable behaviour. For continual learning the problem is rather similar: If we repeatedly optimize the second-order approximation of the loss by taking gradient steps, we will get pushed further and further away from $\mathbf{w}_0$ in directions of negative curvature and thus the network would again forget catastrophically.

1 While this is not necessary, we will only treat classification problems here.

One solution to this problem is to approximate the Hessian by a positive (semi-)definite matrix. One popular choice of such a matrix is the Fisher Information, which is positive semi-definite by design. This can be seen from its definition, which is given by

$$\mathbf{F} = \mathbb{E}_{X\sim\mathcal{X}}\mathbb{E}_{y\sim p(\cdot|X,\mathbf{w})}\left[\mathbf{g}(X,y)\mathbf{g}(X,y)^T\right] \tag{2.2}$$

where $X \sim \mathcal{X}$ is a sample from the input distribution and $y \sim p(\cdot \mid X, \mathbf{w})$ is a sample from the model's output distribution. Notably, this is different from taking the label assigned to X in the dataset. If we do the latter, we obtain a biased approximation of the Fisher Information, known as the "Empirical Fisher" within the machine learning community. See [45, 46] for discussions of this approximation, and [46] for a discussion of differences in naming conventions between statistics and machine learning community. The Empirical Fisher will appear in Chapter 3 and very briefly at the end of Chapter 5.

It is worth noting that the Fisher Information was originally not introduced to machine learning in order to approximate the Hessian, but rather from an information geometric perspective (see e.g. [47]). However, as shown in [45, 48, 49], it often coincides with the Generalised-Gauss-Newton matrix and as such is strongly linked to the Hessian. More precisely, in cases where the network's output distribution over labels y matches the real distribution over labels, the Fisher is equal to the Hessian. Note that in this case the Empirical Fisher also coincides with the Hessian, which perhaps explains partly why the Empirical Fisher can be useful, despite its limitations discussed in [46].

2.1 FISHER FOR CONTINUAL LEARNING

As briefly hinted at in the introduction, the key idea of [11] for continual learning is to replace the loss of a task that has already been learned by the network by its second-order Taylor approximation. This way, we can hope to maintain parameter values $\mathbf{w}$ which have low loss on the old task, even without accessing data from the old task (we only need to access our second-order approximation of the loss).

Thus, to perform continual learning with this idea, we need to repeatedly evaluate the term $\mathbf{w}\mathbf{F}\mathbf{w}^T$ as a proxy for the loss on an old task. Note that the first-order term $\mathbf{w}^T\mathbf{g}$ is usually omitted, since one assumes that the Taylor expansion is taken at (or at least very close to) a local minimum, where $\mathbf{g} \approx 0$.

For completeness note that [11] motivate and derive their algorithm from a Bayesian perspective, so that what we referred to as a second-order Taylor expansion is called the Laplace posterior; but the difference here is only words.

2.2 FISHER FOR OPTIMIZATION

If we use the Fisher for optimization, we need to minimize the expression (2.1) with respect to $\mathbf{w}$. A brief calculation shows that this leads to the update

$$\mathbf{w}^* = \mathbf{F}^{-1}\mathbf{g}$$

In practice, this is not quite the update formula that is used. Rather, one employs

$$\mathbf{w}^* = (\lambda\mathbf{I} + \mathbf{F})^{-1}\mathbf{g}$$

where λ is called a damping term. This can be seen as establishing a trust-region and limiting update size, especially in directions of low or zero curvature (note that the Fisher typically does not have full rank for sufficiently overparametrised networks, as can be seen from (2.2).)

Finally, we point out that this optimization algorithm is known as Natural Gradient Descent (see e.g. [50]). Due to the close link of the Fisher to the Hessian, Natural Gradient Descent is often seen as a second-order optimization algorithm.

2.3 APPROXIMATIONS OF THE FISHER INFORMATION

As stated previously and as can be seen from the definitions, the Fisher (and Hessian) has size $n \times n$, where n is the number of parameters. It is therefore prohibitively expensive to compute and store the entire Fisher matrix for even moderately sized neural networks.

In principle, there are two approaches to circumvent this problem:

(1) In many cases, we are not actually interested in knowing the Fisher. Rather it is an intermediate quantity that is required to compute the final expression of interest. This is true both in continual learning, where we are interested in computing $\mathbf{w}^T\mathbf{F}\mathbf{w}$ and optimization, where we are interested in computing $(\lambda\mathbf{I} + \mathbf{F})^{-1}\mathbf{g}$. In some situations one can evaluate these expressions precisely, without explicitly computing the Fisher Information. We will give two examples for this and point out that both examples only work efficiently under certain additional assumptions, which we will omit

for the sake of brevity and clarity.
The first example is Hessian-Free optimization [39], which exploits that Hessian- (or Fisher)-vector products can be evaluated efficiently in neural networks (without evaluating the Hessian explicitly) and that in conjunction with conjugate gradient descent this is enough to approximate $(\lambda \mathbf{I} + \mathbf{F})^{-1}\mathbf{g}$ reasonably efficiently and precisely.
For the second example, one can exploit the intrinsic low-rank structure of the Fisher Information to compute $(\lambda \mathbf{I} + \mathbf{F})^{-1}\mathbf{g}$ efficiently as was first described in [40]. Since this will be a key tool for Chapter 5, we will outline its main ideas below in Section 2.3.2

(2) As an alternative to not evaluating the Full Fisher explicitly, one can try to approximate it in forms that allow for more memory efficient storage and more efficient evaluation of the desired quantities.
A simple and commonly employed approximation is taking the diagonal entries of the Fisher. This is precisely the approach taken by [11] for continual learning. In optimization, it is interesting that the resulting optimizer is loosely linked to the popular Adam optimizer [51] (but see also [52] for how this link can be made much less handwavy).
Another approximation of the Fisher, which has gained popularity due to its empirical successes in different domains (optimization, continual learning, meta-learning, posterior-approximations), is a block-diagonal, Kronecker-factored approximation, due to [41, 42]. Since this approximation is at the core of Chapter 5, we will explain it in more detail below.

2.3.1 *Block-diagonal, Kronecker-factored Approximation of the Fisher*

The first step of this approximation is to only compute the block-diagonal of the Fisher, where each block corresponds to one layer of the neural network. To make this more precise, we will only care about the (i, j)-th entry of $\mathbf{F}$ when the parameters w_i, w_j come from the same layer of the network.

To derive the Kronecker-factorisation of each block (due to [41, 42]), first interpret the expectation in (2.2) as a sum and then consider each summand individually. Each summand has the form $\mathbf{g}\mathbf{g}^T$[2]. Now, it is well known that for a single layer (for simplicity we assume fully-connected layers only), the gradient $\mathbf{g}$ has "low-rank" structure. To be more precise, if we take the matrix-shaped version of the gradient $\mathbf{g}^{\text{mat}}$ (rather than its vectorised variant), it can be written as a rank 1 matrix $\mathbf{g}^{\text{mat}} = \mathbf{a}\mathbf{e}^T$, where $\mathbf{a}$ is a column vector containing the input activations of the layer (computed in

2 The weight given to each summand by the expectation can be absorbed into $\mathbf{g}$ by rescaling it.

a typical forward pass) and where $\mathbf{e}$ is a column vector containing the derivative of the loss with respect to the output activations of the layer (these derivatives are computed in a typical backward pass). This property is a straightforward consequence of the chain rule.

Luckily, this low rank structure of $\mathbf{g}$ translates to a useful low-rank structure of $\mathbf{g}\mathbf{g}^T$. More precisely, each diagonal block of $\mathbf{g}\mathbf{g}^T$ is given by $\mathbf{a}\mathbf{a}^T \otimes \mathbf{e}\mathbf{e}^T$, where $\otimes$ denotes the Kronecker-product and where the equality follows directly from the definition of the Kronecker-product.

Thus, each diagonal block of the Fisher is a sum (or expectation) over Kronecker-products

$$\mathbf{F}^{\text{Block}} = \sum_i \mathbf{a}^{(i)}\mathbf{a}^{(i)T} \otimes \mathbf{e}^{(i)}\mathbf{e}^{(i)T}$$

where the summation index i simply subsumes the two expectations from (2.2).

To arrive at the final Kronecker-factored approximation of the Fisher, one more trick is performed and we approximate

$$\begin{aligned} \mathbf{F}^{\text{Block}} = & \quad \textstyle\sum_i \mathbf{a}^{(i)}\mathbf{a}^{(i)T} \otimes \mathbf{e}^{(i)}\mathbf{e}^{(i)T} && (2.3) \\ \overset{\propto}{\sim} & \quad \left(\textstyle\sum_i \mathbf{a}^{(i)}\mathbf{a}^{(i)T}\right) \otimes \left(\textstyle\sum_i \mathbf{e}^{(i)}\mathbf{e}^{(i)T}\right) && (2.4) \end{aligned}$$

This last approximation is usually justified by an independence assumption, which is typically violated as discussed briefly in Benzing [44]. It is (in a sense that can be made precise without too much trouble) a rank 1 approximation of the block-diagonal of the Fisher.

The resulting approximation of the Fisher has a few very important properties. (1) It is space efficient: Storing the Kronecker-factors typically requires only about as much space as storing the layers themselves. (2) The required computations can be implemented in a time efficient manner: The desired quantities $\mathbf{a}, \mathbf{e}$ are standard quantities computed during a for- and backward pass, and the summation can be vectorised efficiently. (3) Many downstream quantities, like $\mathbf{w}^T\mathbf{F}\mathbf{w}$ for continual learning or $(\lambda\mathbf{I} + \mathbf{F})^{-1}\mathbf{g}$ for optimization, can be evaluated efficiently. We briefly mention that for performing the inversion $(\lambda\mathbf{I} + \mathbf{F})^{-1}$ there is typically another approximation involved. Since this is not directly related to the Fisher, we defer the unexpectedly important details to Chapter 5.

With the advantages of the Kronecker-factorisation being pointed out, we re-emphasise that evidence presented in Chapter 5 will raise doubt whether using these approximation really preserve the final quantity, i.e. the natural gradient $(\lambda\mathbf{I} + \mathbf{F})^{-1}\mathbf{g}$ faithfully or whether the approximations in fact (and

somewhat accidentally) pick up on other relevant local properties of the loss landscape.

2.3.2 *Low Rank Fisher and Efficient Subsampled Natural Gradients*

As pointed out above, in some situations we can evaluate the natural gradient

$$(\lambda \mathbf{I} + \mathbf{F})^{-1}\mathbf{g}$$

efficiently and exactly.

The fairly simple ideas to do so were first described in [40, 53] and later rediscovered by us. Here, we provide a sketch of these ideas, full details can be found in Appendix I of Benzing [44], which makes some modest improvements over [40].

To see how the natural gradient can be evaluated efficiently in some situations, we briefly restate the definition of the Fisher

$$\mathbf{F} = \mathbb{E}_{X\sim\mathcal{X}}\mathbb{E}_{y\sim p(\cdot|X,\mathbf{w})}\left[\mathbf{g}(X,y)\mathbf{g}(X,y)^T\right] \tag{2.5}$$

and note that (again, viewing the expectation as a sum), the Fisher is explicitly given as a sum over rank-1 matrices. In particular, if not many summands appear, the Fisher has low rank and we can use the matrix inversion lemma to invert $(\lambda \mathbf{I} + \mathbf{F})$.

To make this concrete, write $\mathbf{F} = \mathbf{G}\mathbf{G}^T$, where $\mathbf{G}$ is an $n \times r$ matrix. Then by the matrix inversion lemma

$$(\lambda \mathbf{I} + \mathbf{F})^{-1} = (\lambda \mathbf{I} + \mathbf{G}\mathbf{G}^T)^{-1} = \lambda^{-1}\mathbf{I} - \lambda^{-2}\mathbf{G}\Big(\mathbf{I} + \frac{1}{\lambda}\mathbf{G}^T\mathbf{G}\Big)^{\cdot^{-1}}\mathbf{G}^T \tag{2.6}$$

Thus, we have solved the first problem, namely we do not need to invert a large $n \times n$ matrix, but only the $r \times r$ matrix $\mathbf{G}^T\mathbf{G}$. Luckily, this matrix can be evaluated efficiently[3].

The next problem, which is that we cannot evaluate the expression (2.6) explicitly (due to its prohibitively large size), can also be overcome. We are only interested in computing the matrix-vector product $(\lambda \mathbf{I} + \mathbf{F})^{-1}\mathbf{g}$, and to do so we can use the RHS of (2.6) and first evaluate $\mathbf{G}^T\mathbf{g}$, then evaluate $\mathbf{v} = (\mathbf{I} + \frac{1}{\lambda}\mathbf{G}^T\mathbf{G})^{-1}(\mathbf{G}^T\mathbf{g})$ and finally compute $\mathbf{G}\mathbf{v}$. To see that steps one and three in this procedure can be done efficiently, note that $\mathbf{G}^T\mathbf{g} = (\mathbf{g}^T\mathbf{G})^T$

3 Each entry of $\mathbf{G}^T\mathbf{G}$ is a gradient-gradient dot-product, which can be computed efficiently using the low rank strucutre of gradients mentioned above, and allows for vectorisation of computing all entries of $\mathbf{G}^T\mathbf{G}$, see e.g. Benzing [44].

is a vector-jacobian-product and that $\mathbf{Gv}$ is a jacobian-vector-product and both can be done efficiently.
To summarise, we can evaluate natural gradients efficiently, provided that r, the number of rank-1 summands forming the Fisher, is not too large.

For classification problems with c class labels and d input vectors, the Fisher has rank cd. Thus, since cd is usually large for standard machine learning datasets, it typically is still infeasible to perform the above computations. Nevertheless, there are two advantages to the described procedure, which will both be exploited in Chapter 5: (1) On small datasets, we can evaluate natural gradients exactly. (2) On large dataset, we can compute the Fisher on a subset of images (known as the subsampled Fisher) and perform exact computations for the subset. While this is of course not the same as exact natural gradients, it can be used in control settings to compare to other natural gradient algorithms.

3

UNIFYING REGULARISATION METHODS FOR CONTINUAL LEARNING

This chapter is taken and partially adapted from Benzing [54], which was presented as a long oral talk at AISTATS 2022.

3.1 INTRODUCTION

As explained in Chapter 1, one key limitation of gradient based learning is that it overwrites previously acquired knowledge when different tasks are learned sequentially. We will soon provide a formal definition of the exact setup.

By now a lot (if not the majority) of new work on continual learning relies fairly heavily on replay. Roughly speaking, this means that a subset of data from previous task is stored and replayed when learning new tasks. The predominance of these methods is perhaps explained by their empirical success and the community's focus on benchmark numbers: Algorithms that do not use replay buffers, or only do so in a very limited way, struggle competing with replay methods, as has been noted early on and many times (e.g. [55, 56]).

There is certainly a case to be made for investigating replay methods as they might be feasible solutions in real-world engineering settings and enhance our understanding of which parts or examples of a distribution are particularly informative. Nevertheless, investigating and improving methods which do not use replay remains an important question, both from a scientific point of view and in some applications, where storing old examples might be strictly disallowed (for example due to privacy constraints). This holds true even if, or perhaps especially when, algorithms in the no-replay domain struggle to compete with replay methods. As we have also mentioned in Chapter 1 progress on this problem may improve our understanding of the loss landscape of neural networks, which in turn could have implications across a range of areas, like bayesian ML, optimization or transfer/meta-learning.

With this in mind, this chapter will focus on regularisation methods for continual learning which are an important line of work for contin-

ual learning without replay. The first regularisation-based method was introduced by Kirkpatrick *et al.* [11], who proposed the Elastic Weight Consolidation (EWC) algorithm. After training a given task, EWC measures the importance of each parameter for this task and introduces an auxiliary loss penalising large changes in important parameters. Naturally, this raises the question of how to measure 'importance'. While EWC uses the diagonal Fisher Information, two main alternatives have been proposed: Synaptic Intelligence (SI, [13]) aims to attribute the decrease in loss during training to individual parameters and Memory Aware Synapses (MAS, [14]) introduces a heuristic measure of output sensitivity. Together, these three approaches have inspired many further regularisation-based approaches, including combinations of them [17], refinements [15–18], extensions [19–22] and applications in different continual learning settings [57] as well as different domains of machine learning [58]. Almost every new continual learning method compares to at least one of the algorithms EWC, SI and MAS.

Despite their influence, basic practical and theoretical questions regarding these algorithms had previously been unanswered. Notably, it was unknown how similar these importance measures are. Additionally, for SI and MAS as well as their follow-ups there was no solid theoretical understanding of their effectiveness. Here, we close both these gaps through a theoretical analysis confirmed by a series of carefully designed experiments on standard continual learning benchmarks. Our main findings in this chapter can be summarised as follows:

(a) We show that SI's importance approximation is biased and that the bias rather than SI's original motivation is responsible for its performance. Further, also due to the bias, SI is approximately equal to the square root of the Fisher Information.

(b) We show that MAS, like SI, approximately equals the square root of the Fisher Information.

(c) Together, (a) and (b) unify the three main regularisation approaches and their follow ups by explicitly linking all of them to the same theoretically justified quantity – the Fisher Information. For SI- and MAS-based algorithms this has the additional benefit of giving a more plausible theoretical explanation for their effectiveness.

(d) Based on our precise understanding of SI, we propose an improved algorithm, Second-Order Synapses (SOS). We demonstrate that SOS outperforms SI in various regimes.

3.2 REVIEW OF REGULARISATION METHODS AND RELATED WORK

Here, we review importance based regularisation approaches (but see also [59] for non importance based regularisation). Additional related work is discussed in Supplementary 3.7.2.

Formal Description of Continual Learning.
In continual learning we are given K datasets $\mathcal{D}_1, \ldots, \mathcal{D}_K$ sequentially. When training a neural net with N parameters $\mathbf{w} \in \mathbb{R}^N$ on dataset $\mathcal{D}_k$, we have no access to the previously seen datasets $\mathcal{D}_{1:k-1}$. However, at test time the algorithms is tested on all K tasks and the average accuracy is taken as measure of the performance.

Common Framework for Regularisation Methods.
Regularisation based approaches introduced in [11] (with some slight adaptations proposed in [15]) protect previous memories by modifying the loss function L_k related to dataset $\mathcal{D}_k$. Let us denote the parameters obtained after finishing training on task k by $\mathbf{w}^{(k)}$ and let $\boldsymbol{\lambda}^{(k)} \in \mathbb{R}^N$ be the parameters' *importances*. When training on task k, regularisation methods use the loss

$$\tilde{L}_k = L_k + c \cdot \sum_{i=1}^{N} \lambda_i^{(k-1)} \left(w_i - w_i^{(k-1)} \right)^2$$

where $c > 0$ is a hyperparameter. The first term L_k is the standard (e.g. cross-entropy) loss of task k. The second term ensures that the parameters do not move away too far from their previous values. In some cases, the auxiliary loss can be interpreted directly as modelling the previous tasks' losses $L_1 + \ldots + L_{k-1}$ by their second-order Taylor expansion.

Usually, $\lambda_i^{(k)} = \lambda_i^{(k-1)} + \lambda_i$, where λ_i is the importance for the most recently learned task k, and this is also the update prescribed by the second-order approximation view mentioned above.

Elastic Weight Consolidation
[11] uses the diagonal of the Fisher Information as importance measure and in this case the auxiliary regularisation loss can be interpreted as an approximation of the second-order Taylor expansion of the loss of previous tasks as mentioned above. For an equivalent bayesian motivation, see [11, 15].

We briefly restate and slightly rewrite the definition of the Fisher for later purposes

$$\mathrm{F} = \mathbb{E}_{X\sim\mathcal{X}}\mathbb{E}_{y\sim\mathbf{p}_X}\left[\mathbf{g}(X,y)\mathbf{g}(X,y)^T\right] \tag{3.1}$$

$$= \mathbb{E}_{X\sim\mathcal{X}}\sum_{y\in L}\mathbf{p}_X(y)\cdot\mathbf{g}(X,y)\mathbf{g}(X,y)^T. \tag{3.2}$$

Note that the Fisher is evaluated at the end of each task, to obtain a taylor expansion around the final point of the optimization trajectory.
To make things concrete, taking only the **diagonal Fisher** means $\lambda_i(\mathrm{EWC}) = \mathbb{E}_{X\sim\mathcal{X}}\mathbb{E}_{y\sim\mathbf{p}_X}\left[g_i(X,y)^2\right]$.

Variational Continual Learning
[60] shares its Bayesian motivation with EWC, but uses principled variational inference in Bayesian neural networks. Its similarity to EWC and the fact that it uses 'a smoothed version of the Fisher' is already discussed in [60] and further formalised in [61]. We will therefore focus on the two algorithms below.

Memory Aware Synapses
[14] heuristically argues that the sensitivity of the function output with respect to a parameter should be used as the parameter's importance. This sensitivity is evaluated after training a given task. It can be measured with respect to either the logits or the probabilities. Here, we choose the latter option and describe the precise importance this leads to. In Appendix H of Benzing [54] we show that the resulting version of MAS has the same performance as the one described here, and, crucially, is also related to the Fisher Information.
Denoting, as before, the final layer of learned probabilities by $\mathbf{p}_X$, the MAS importance is

$$\lambda_i(\mathrm{MAS}) = \mathbb{E}_{X\sim\mathcal{X}}\left[\left|\frac{\partial\|\mathbf{p}_X\|^2}{\partial w_i}\right|\right],$$

MAS suggests using validation data to measure this importance. For a fair comparison, we measure importances on training data for all methods. We also confirmed that this choice does not affect our results.

Synaptic Intelligence
[13] approximates the contribution of each parameter to the decrease in loss and uses this contribution as importance. To formalise the 'contribution of a parameter', let us denote the parameters at time t by $\mathbf{w}(t)$ and the loss by $L(t)$. If the parameters follow a smooth trajectory in parameter space, we

write $\mathbf{w}'$ for the temporal derivative of the parameters, and we can rewrite the *decrease* in loss from time 0 to T as

$$L(0) - L(T) = -\int_{\mathbf{w}(0)}^{\mathbf{w}(T)} \frac{\partial L(t)}{\partial \mathbf{w}} \mathbf{w}'(t) dt \tag{3.3}$$

$$= -\sum_{i=1}^{N} \int_{w_i(0)}^{w_i(T)} \frac{\partial L(t)}{\partial w_i} w_i'(t) dt. \tag{3.4}$$

The i-th summand in (3.4) can be interpreted as the contribution of parameter w_i to the decrease in loss. While we cannot evaluate the integral exactly, we can use a first-order approximation to obtain the importances. To do so, we write $\Delta_i(t) = (w_i(t+1) - w_i(t))$ for an approximation of $w_i'(t)dt$ and get

$$\tilde{\lambda}_i(\mathrm{SI}) = \sum_{t=0}^{T-1} \frac{\partial L(t)}{\partial w_i} \cdot \Delta_i(t). \tag{3.5}$$

Thus, we have $\sum_i \tilde{\boldsymbol{\lambda}}_i(SI) \approx L(0) - L(T)$. In addition, SI rescales its importances as follows[1]

$$\lambda_i(\mathrm{SI}) = \frac{\max\left\{0, \tilde{\lambda}_i(\mathrm{SI})\right\}}{(w_i(T) - w_i(0))^2 + \xi}. \tag{3.6}$$

Zenke, Poole & Ganguli [13] use additional assumptions to justify this importance. One of the them – using full batch gradient descent – is violated in practice and we will show that this has an important consequence.

Additional regularisation approaches.
EWC, SI and MAS have inspired several follow ups. We presented a version of EWC due to [15] and tested in [17, 19]. It is theoretically more sound and was shown to perform better. [17] combine EWC with a 'KL-rescaled'-SI. [16] use a block-diagonal (rather than diagonal) approximation of the Fisher; [18] use the full Hessian matrix for small networks. [20] rotate the network to diagonalise the most recent Fisher Matrix, [21] modify the loss of SI and [22] aim to account for batch-normalisation.

Overview over algorithms
Below we will design and introduce a few variants of the algorithms described above to test a set of different hypothesis. For reference and readability a tabular overview over these variants is provided in Supplementary Table 3.3.

1 Note that the $\max(0, \cdot)$ is not part of the description in [13]. However, we needed to include it to reproduce their results. A similar mechanism can be found in the official SI code.

3.3 SYNAPTIC INTELLIGENCE, ITS BIAS AND THE FISHER

Here, we explain why SI is approximately equal to the square root of the Fisher, despite the apparent contrast between the latter and SI's path integral. First, we identify the bias of SI when approximating the path integral. We then show that the bias, rather than the path integral, explains similarity to the Fisher as well as performance. We carefully validate each assumption of our analysis empirically.

3.3.1 *Bias of Synaptic Intelligence*

To calculate $\lambda(\text{SI})$ (3.5), we need to calculate the product

$$p = \frac{\partial L(t)}{\partial w} \cdot \Delta(t)$$

for each t. Evaluating the full gradient $\frac{\partial L}{\partial w}$ is too expensive, so SI uses a stochastic minibatch gradient. The estimate of p is biased since the same minibatch is used for the update Δ and the estimate of $\partial L/\partial w$.

We now give the calculations detailing this argument. For ease of exposition, let us assume vanilla SGD with learning rate 1 is used. Given a minibatch, we slightly change notation and denote its stochastic gradient estimate by $g + \sigma$, where $g = \partial L/\partial w$ denotes the full gradient and σ the noise. The update is $\Delta = g + \sigma$. Thus, p should be

$$p = g \cdot (g + \sigma).$$

However, using $g + \sigma$, which was used for the parameter update, to estimate g results in

$$p_{bias} = (g + \sigma)^2.$$

Thus, the gradient noise introduces a bias of

$$\mathbb{E}[\sigma^2 + \sigma g] = \mathbb{E}[\sigma^2].$$

Unbiased SI (SIU)
Having understood the bias, we can design an unbiased estimate by using two independent minibatches to calculate Δ and estimate g. We get $\Delta = g + \sigma$ and an estimate $g + \sigma'$ for g with independent noise σ'. We obtain

$$p_{no\_bias} = (g + \sigma') \cdot (g + \sigma)$$

which in expectation equals $p = g \cdot (g + \sigma)$. Based on this we define an unbiased importance measure

$$\tilde{\lambda}_i(\text{SIU}) = \sum_{t=0}^{T-1} (g_t + \sigma'_t) \cdot \Delta(t).$$

Bias-Only version of SI (SIB)
To isolate the bias, we can take the difference between biased and unbiased estimate. Concretely, this gives an importance which only measures the bias of SI

$$\tilde{\lambda}_i(\text{SIB}) = \sum_{t=0}^{T-1} ((g + \sigma) - (g + \sigma'_t)) \cdot \Delta(t).$$

Observe that this estimate multiplies the parameter update $\Delta(t)$ with nothing but stochastic gradient noise. From the perspective of the SI path-integral, this should be meaningless and perform poorly. Our theory, detailed below, predicts differently.

3.3.2 *Relation of SI's Bias to Fisher*

The bias of SI depends on the optimizer used. The original SI-paper (and we) uses Adam [51] and we now analyse the influence of this choice in detail; for other choices see Supplementary 3.7.1.

Recall that $\tilde{\lambda}(SI)$ is a sum over terms $\frac{\partial L(t)}{\partial w} \cdot \Delta(t)$,
where $\Delta(t) = w(t+1) - w(t)$ is the parameter update at time t. Both terms, $\frac{\partial L(t)}{\partial w}$ as well as $\Delta(t)$, are computed using the same minibatch. Given a stochastic gradient $g_t + \sigma_t$, Adam keeps an exponential average of the gradient

$$m_t = (1 - \beta_1)(g_t + \sigma_t) + \beta_1 m_{t-1}$$

as well as the gradient

$$v_t = (1 - \beta_2)(g_t + \sigma_t)^2 + \beta_2 v_{t-1}.$$

Ignoring minor normalisations, the parameter update is

$$\Delta(t) = \eta_t m_t / \sqrt{v_t},$$

with learning rate $\eta_t = 0.001, \beta_1 = 0.9$ and $\beta_2 = 0.999$. Thus,

$$\frac{\partial L(t)}{\partial w}\Delta(t) = \eta_t(1-\beta_1)\frac{(g_t+\sigma_t)^2}{\sqrt{v_t}} + \eta_t\beta_1\frac{(g_t+\sigma_t)m_{t-1}}{\sqrt{v_t}}$$
$$\overset{\textbf{(A1)}}{\approx} \eta_t(1-\beta_1)\frac{(g_t+\sigma_t)^2}{\sqrt{v_t}} \tag{3.7}$$

Here, we made **Assumption (A1)** that the gradient noise is larger than the gradient, or more precisely:

$$(1-\beta_1)\sigma_t^2 \gg \beta_1 m_{t-1} g_t$$

(we ignore $\sigma_t m_{t-1}$ since $\mathbb{E}[\sigma_t m_{t-1}] = 0$ and since we average SI over many time steps). **(A1)** is equivalent to the bias of SI being larger than its unbiased part as detailed in Supplementary 3.7.5. Experiments in Section 3.3.3 and Supplementary 3.7.4 provide strong support for this assumption. Next, we rewrite

$$\tilde{\lambda}(\text{SI}) \overset{\textbf{(A1)}}{\approx} (1-\beta_1)\sum_{t\leq T}\frac{\eta_t(g_t+\sigma_t)^2}{\sqrt{v_t}} \tag{3.8}$$
$$= \frac{1-\beta_1}{\sqrt{v_T}}\sum_{t\leq T}\eta_t\sqrt{\frac{v_T}{v_t}}(g_t+\sigma_t)^2 \tag{3.9}$$

Now consider v_t. It is a decaying average of $(g_t+\sigma_t)^2$. When stochastic gradients become smaller during learning, v_t will decay so that

$$\sum \sqrt{v_T/v_t}\cdot(g+\sigma_t)^2$$

will be a (unnormalised, not necessarily exponentially) decaying average of the squared gradient. Therefore, we make **Hypothesis (A2)**

$$\sum_{t\leq T}\sqrt{\frac{v_T}{v_t}}\cdot(g+\sigma_t)^2 \overset{\textbf{(A2)}}{\propto} v_T,$$

since both LHS & RHS are decaying averages of the squared gradient. We will validate **(A2)** in Section 3.3.3. Altogether, we obtain

$$\tilde{\boldsymbol{\lambda}}(\text{SI}) \approx \frac{1-\beta_1}{\sqrt{v_T}}\sum_{t\leq T}\eta_t\sqrt{\frac{v_T}{v_t}}(g_t+\sigma_t)^2 \tag{3.10}$$
$$\overset{\textbf{(A2)}}{\propto} \frac{v_T}{\sqrt{v_T}} = \sqrt{v_T}.$$

No.	Algo.	P-MNIST	CIFAR
(1)	SI	97.2±0.1	74.4±0.2
	SI Bias-Only	97.2±0.1	75.1±0.1
	SI Unbiased	96.3±0.1	72.5±0.3
	SOS	97.3±0.1	74.1±0.2
(2)	SI(2048)	96.2±0.1	70.0±0.3
	SOS(2048)	**97.1 ± 0.1**	**74.4 ± 0.1**
(3)	MAS	97.3±0.1	73.7±0.2
	AF	97.4±0.1	73.4±0.1
	$\sqrt{\text{Fisher}}$	97.1±0.2	73.5±0.2
	Fisher (EWC)	97.1±0.2	73.1±0.2

TABLE 3.1: **Test Accuracies on Permuted-MNIST and Split CIFAR** (Mean and std-err of average accuracy over 3 resp. 10 runs for MNIST resp. CIFAR). Experiments No. (1) & (3) explain SI's and MAS' performance. Exp (2) shows improvements of our SOS for large batch training and confirms our theoretical prediction.

To avoid confusion, we note that it may seem at first that a similar argument implies that SI is proportional to v_t rather than $\sqrt{v_t}$. Appendix D in Benzing [54] explains why this is not the case.

Relation of SI to Fisher (EWC).
Recall that the exact form of the importance of SI depends on the optimizer used. Here, we used Adam following [13]. The effect of SGD (+momentum) is very similar as discussed in Supplementary 3.7.1.
With Adam, the importance of SI is (due to its bias) roughly equal to $\sqrt{v_t}$. Note that v_t is closely related to the Empirical Fisher, especially when the batch size is not too big (see Supplementary 3.7.3). Thus, other than using different approximations of the Fisher, the only difference between EWC and SI is the square root of the latter.
Even though we are not aware of a theoretical justification of the square root in this setting, this result explicitly links the bias and importance measure of SI to second order information of the Fisher. On a related note, we show in Appendix G of Benzing [54] that in a different regime the square root of the Hessian can be a theoretically justified importance measure.

Influence of Regularisation.
Note that gradients, second moment estimates and the update direction $\Delta(t)$ of Adam will be influenced by the auxiliary regularisation loss. In contrast, the approximation of $\partial L/\partial w$ in (3.7) only relies on current task gradients (without regularisation). Thus, the relation between SI and $\sqrt{v_T}$ (which in this context is decaying average of task-only gradients), will be more noisy in the presence of large regularisation. This will be confirmed empirically in Section 3.3.3.

Practical Implications.
A benefit of our derivation is that it makes the dependence of SI on the optimisation process explicit. Consider for example the influence of batch size. A priori one would not expect it to affect SI more than other methods. Given our derivation, however, one can see that a larger batch size reduces noise and thus bias, making **(A1)** less valid. Moreover, v_t will be a worse approximation of the Fisher (see Supplementary 3.7.3 or e.g. [62]). Thus, if SI relies on its relation to second order information of the Fisher, then we expect it to suffer disproportionately from large batch sizes – a prediction we will confirm empirically below. Additionally, learning rate decay, choice of optimizer and trainingset size and difficulty could harm SI as discussed in more detail in Supplementary 3.7.1.

Improving SI: Second-Order Synapses (SOS)
Given our derivation, we propose to adapt SI to explicitly measure second order information contained in the Fisher. To this end, we propose the algorithm Second-Order Synapses (SOS). In its simplest form it uses $\boldsymbol{\lambda}(\text{SOS}) = \sqrt{v_T}$ as importance, where v_t is evaluated as described in the beginning of Section 3.3.2. Note that v_t can be measured independently of which optimizer is used. The value of v_t at the end of tranining a given task is taken as the algorithms importance. Importances from different tasks are summed, analogously to the other regularisation methods.

Moreover, for larger batch sizes we introduce a similar, but provably better approximation of the Fisher detailed in Supplementary 3.7.3. We will confirm some benefits of SOS over SI empirically below and discuss additional advantages in Supplementary 3.7.1. Note also that SOS is computationally more efficient than MAS and EWC as it does not need a pass through the data after training a given task.

Model	**SI**	**SOS**	**EWC***	**MAS***
Small	25.1 ± 4.6	44.3 ± 0.1	45.1	40.6
Base	46.0 ± 0.1	43.3 ± 0.3	42.4	46.9
Wide	40.0 ± 0.2	46.0 ± 0.1	31.1	45.1
Deep	21.6 ± 0.7	30.0 ± 0.1	29.1	33.6

TABLE 3.2: **Test Accuracies on TinyImagenet.** Algorithms SI and SOS are evaluated on a version of ImageNet on different VGG architectures; results averaged across 3 seeds. Framework, hyperparameter selection and code directly taken from [63] and only modified to support SOS. Results for EWC, MAS marked with an asterisk "*" are reported directly from [63] without independent replication. Our replication of SI performs better than reported in [63], despite using the original code. SOS has substantial performance gains over SI.

3.3.3 *Empirical Investigation of SI, its Bias and Fisher*

Bias dominates SI

According to the motivation of SI the sum of importances over parameters $\sum_i \tilde{\lambda}_i(\mathrm{SI})$ should track the decrease in loss $L(0) - L(T)$, see (3.4). Therefore, we investigated how well the summed importances of SI and its unbiased version SIU approximate the decrease in loss. We also include an approximation of the path integral which uses the full training set gradient $\partial L / \partial w$. The results in Figure 3.1 (left) show: (1) Using an unbiased gradient estimate and the full gradient gives almost identical sums, supporting the validity of the unbiased estimator.[2] (2) The bias of SI is 5-6 times larger than its unbiased component so that SIU yields a considerably better approximation of the path integral.

Checking (A1)

We point out that the bias, i.e. the difference between SI and SIU, in Figure 3.1 (left) is due to the term $(1 - \beta_1)\sigma_t^2$. Thus, the fact that the bias is considerably larger then the unbiased part is direct, strong evidence of Assumption **(A1)** (see Supplementary 3.7.5 for full calculation). Results on CIFAR are analogous, see Benzing [54].

2 Note that even the unbiased first order approximation of the path integral overestimates the decrease in loss. This is consistent with findings that the loss has positive curvature [58, 64].

Checking (A2): SI is almost equal to the Square Root of Approximate Fisher v_t.
To check **(A2)**, we compared $\lambda(\text{SI})$ and $\lambda(\text{SOS}) = \sqrt{v_t}$, see Figure 3.1 (mid, right). Correlations are almost equal to 1 on MNIST. The same holds on CIFAR Task 1, where the slight drop in correlation from around 0.99 to 0.9 is only due to the division in equation (3.6) (c.f. Appendix L of Benzing [54]). In summary, this shows that $\sqrt{v_t}$ is a very good approximation of SI. Correlations of SI to SOS on CIFAR Task 2-6 decrease due to regularisation, see below.

Effect of Regularisation
The drop of correlations on CIFAR tasks 2-6 is due to large regularisation as explained theoretically in Section 3.3.2 and confirmed by two controls of SI with less strong regularisation: The first control simply sets regularisation strength to $c = 0$. The second control refrains from re-initialising the network weights after each task (exactly as in original SI, albeit with slightly worse validation performance than the version with re-initialisation). In the second setting the current parameters $\mathbf{w}$ never move too far from their old value $\mathbf{w}^{(k-1)}$, implying smaller gradients from the quadratic regularisation loss, and also meaning that a smaller value of $c = 0.5$ is optimal. We see that for both controls with weaker regularisation the correlation of SI to SOS is again close to one, supporting our theory.

Bias explains SI's performance
We saw that SI's bias is much larger than its unbiased part. But how does it influence SI's performance? To quantify this, we compared SI to its unbiased version SIU and the bias-only version SIB. Note that SIB is completely independent of the path integral motivating SI, only measures gradient noise and therefore should perform poorly according to SI's original motivation. However, the results in Table 3.1(1) reveal the opposite: Removing the bias reduces performance of SI (SIU is worse), whereas isolating the bias does not affect or slightly improve performance. This demonstrates that SI relies on its bias, and not on the path integral, for its continual learning performance.

Bias explains Second-Order Information.
We have seen that the bias, not the path integral, is responsible for SI's performance. Our theory offers an explanation for this surprising finding, namely that the bias is responsible for SI's relation to the Fisher. But does this explanation hold up in practice? To check this, we compare SI, its unbiased variant SIU and its bias-only variant SIB to the second order

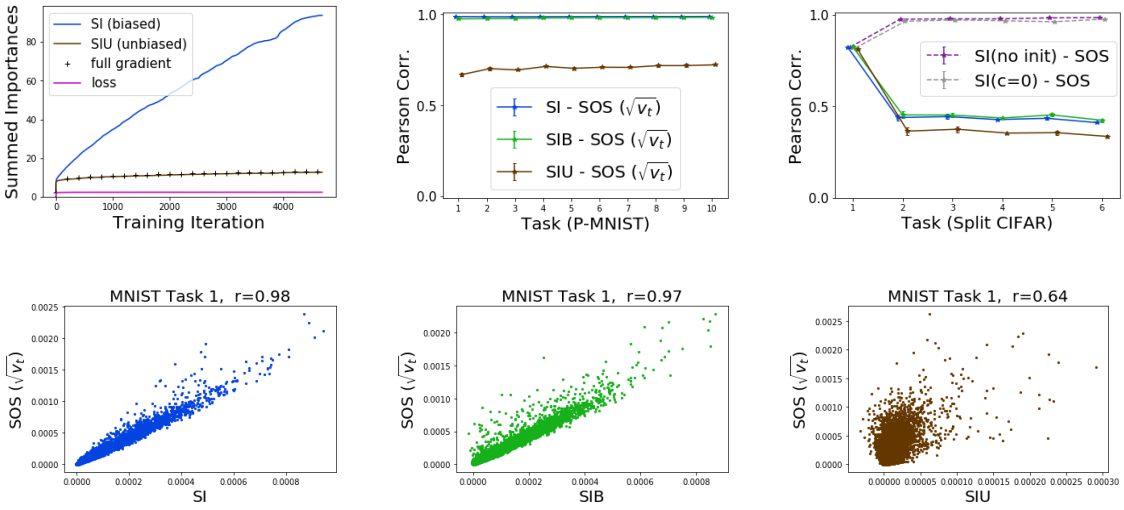


FIGURE 3.1: **SI, Bias and Square Root of Approximate Fisher v_t (SOS).**
Top Left: Summed Importances for SI and its unbiased version, showing that the bias dominates SI and that Assumption **(A1)** holds.
Top Center: Pearson correlations of SI, its bias (SIB), and unbiased version (SIU) with SOS, showing that relation between SI and SOS is strong and due to bias; confirming **(A2)**. The same is shown by the scatter plots.
Top Right: Same as Mid but on CIFAR; additionally, relation between SOS and two SI-controls is shown: 'no init' does not re-initialise network weights after each task; '$c = 0$' has regularisation strength 0. This shows that strong regularisation weakens the tie between SI and SOS on Tasks 2-6 as explained by our theory.
Bottom: Scatter plots of SOS ($\sqrt{v_t}$) with SI (left), its bias-only SIB (middle) and its unbiased version SIU (right); showing 10^5 randomly sampled weights. A straight line through the origin corresponds to two importance measures being multiples of each other as was suggested by our derivation for SI and SOS, but not for SIU and SOS. Note that SIU has negative importances before rescaling in (3.6).

information $\lambda(\text{SOS}) = \sqrt{v_T}$. The results in Figure 3.1 (top middle&right; bottom) show that SI, SIB are more similar to $\sqrt{v_T}$ than SIU, directly supporting our theoretical explanation.

Together the two previous paragraphs are very strong evidence that SI relies on the second order information contained in its bias rather than on the path integral.

SOS improves SI

We predicted that SI would suffer disproportionately from training with large batch sizes. To test this prediction, we ran SI and SOS with larger batch sizes of 2048, Table 3.1 (2). We found that, indeed, SI's performance degrades by roughly 1% on MNIST and more drastically by 4% on CIFAR. In contrast, SOS' performance remains stable.[3] This validates our prediction and demonstrates that our improved understanding of SI leads to notable performance gains.

In addition, we carried out experiments on a version of ImageNet, following the protocol of a recent large-scale comparison of continual learning algorithms [63]. The results are shown in Table 3.2. There is one setting (Base), where SI is slightly better than SOS and we are unsure why. Overall SOS clearly outperforms SI. This further supports the view that explicitly relying on the second-order information of the Fisher provides a more reliable, better performing algorithm.

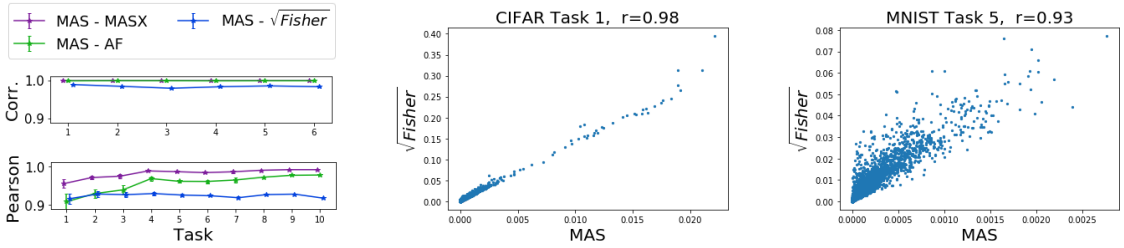


FIGURE 3.2: **Empirical Relation between MAS and Square Root of Fisher.**
Left: Summary of Pearson Correlations (top: CIFAR, bottom: MNIST), supporting Assumptions **(B1)-(B3))**.
Mid & Right: Scatter plots of importance measures. Each point in the scatter plot corresponds to one weight of the net, showing 10^5 randomly sampled weights. A straight line through the origin corresponds to two importance measures being multiples of each other as was suggested by our theoretical analysis.

3.4 MEMORY AWARE SYNAPSES (MAS) AND FISHER

Here, we explain why – despite its different motivation and like SI – MAS is approximately equal to the square root of the Fisher Information. We do so by first relating MAS to AF $:= \mathbb{E}[|g + \sigma|]$ and then theoretically

3 We emphasise that the difference of SI and SOS is *not* explained by a change in 'training regime' [65], since both algorithms use equally large minibatches.

showing how this is related to the Fisher $F = \mathbb{E}[|g+\sigma|^2]$ under additional assumptions. As before, we check our theoretical derivations empirically.

3.4.1 *Theoretical Relation of MAS and Fisher*

We take a closer look at the definition of the importance of MAS. Recall that we use the predicted probability distribution of the network to measure sensitivity rather than logits. With linearity of derivatives, the chain rule and writing $y_0 = \text{argmax}\mathbf{p}_X$, we see (omitting expectation over $X \sim \mathcal{X}$)

$$\begin{aligned}\left|\frac{\partial\|\mathbf{p}\|^2}{\partial\mathbf{w}}\right| &= 2\left|\sum_{y\in Y}\mathbf{p}(y)\frac{\partial\mathbf{p}(y)}{\partial\mathbf{w}}\right| \overset{\text{(B1)}}{\approx} 2\left|\mathbf{p}(y_0)\frac{\partial\mathbf{p}(y_0)}{\partial\mathbf{w}}\right| \\ &= 2\mathbf{p}^2(y_0)\left|\frac{\partial\log\mathbf{p}(y_0)}{\partial\mathbf{w}}\right| \end{aligned} \tag{3.11}$$

Here, we made **Assumption (B1)** that the sum is dominated by its maximum-likelihood label y_0, which should be the case if the network classifies its images confidently, i.e. $\mathbf{p}(y_0) \gg \mathbf{p}(y)$ for $y \neq y_0$. Recall that the importance is measured at the end of training, so that this assumption is justified if the task has been learned successfully. Using the same assumption (B1) we get

$$\mathbb{E}_{y\sim\mathbf{p}_X}\left[|g+\sigma|\right] \overset{\text{(B1)}}{\approx} \mathbf{p}(y_0)|g+\sigma| = \mathbf{p}(y_0)\left|\frac{\partial\log\mathbf{p}(y_0)}{\partial\mathbf{w}}\right|.$$

showing that the only difference between $\lambda(\text{MAS})$ and $\mathbb{E}[|g+\sigma|]$ is a factor of $2\mathbf{p}_X(y_0)$. It is reasonable to make **Assumption (B2)** that this factor is approximately constant, since the model learned to classify training images confidently. This leads to **(B2)**: $\boldsymbol{\lambda}(\text{MAS}) \propto \text{AF} = \mathbb{E}[|g|]$. Note that even with a pessimistic guess that $\mathbf{p}_X(y_0)$ is in a range of 0.5 (rather inconfident) and 1.0 (absolutely confident), the two measures would be highly correlated.

Now, it remains to explore how $\mathbb{E}[|g+\sigma|]$ is related to the Fisher $\text{F} = \mathbb{E}[|g+\sigma|^2]$. The precise relationship between the two will depend on the distribution of $g+\sigma$. If, for example, gradients are distributed normally $(g+\sigma) \sim \mathcal{N}(\mu,\Sigma)$ with $\Sigma_{i,i} \gg \mu_i$ (corresponding to the observation that the noise is much bigger than the gradients, which we have seen to be true above), then we obtain $\text{F}_i \approx \Sigma_{i,i}$ and, since $|g+\sigma|$ follows a folded normal distribution, we also have $\mathbb{E}[|g+\sigma|] \approx c\sqrt{\Sigma_{ii}}$ with $c = \sqrt{2/\pi}$. Thus, in the case of a normal distribution with large noise, we have $\text{AF} = \mathbb{E}[|g|] \propto \sqrt{\text{F}}$. Note that the same conclusion arises in under different assumptions, too, e.g. assuming that gradients follow a Laplace distribution with *scale* $\gg$ *mean*.

Relation of Fisher and MAS
The above analysis leads to **Hypothesis (B3)**: $\boldsymbol{\lambda}(\text{MAS}) \propto \sqrt{\text{Fisher}}$.

3.4.2 *Empirical Relation of MAS and Fisher*

First, to assess **(B1)** in (3.11) we compare LHS (MAS) and RHS (referred to as MASX) at the end of each task. We consistently find that the pearson correlations are almost equal to 1 strongly supporting **(B1)**, see Figure 3.2.

The correlations between $\lambda(\text{MAS})$ and $\text{AF} = \mathbb{E}[|g + \sigma|]$ are similarly high, confirming **(B2)**.

The correlations between $\boldsymbol{\lambda}(\text{MAS})$ and the square root of the Fisher $\sqrt{\text{F}}$ are around 0.9 for Permuted MNIST and approximately 1 for all CIFAR task. This confirms our theoretical hypothesis **(B3)** that MAS is approximately equal to the square root of the Fisher.

In addition, to check whether similarity to the square root of the Fisher can serve as an explanation for performance of MAS, we ran continual learning algorithms based on MAS, AF and $\sqrt{\text{F}}$. The fact that these algorithms perform similarly, Table 3.1 (3), is in line with the claim that similarity to $\sqrt{\text{F}}$ provides a theoretically plausible explanation for MAS' effectiveness.

3.5 EXPERIMENTAL SETUP

We outline the experimental setup, see Appendix F of Benzing [54] for full details. It closely follows SI's setting [13]: In **domain-incremental Permuted MNIST** [10] each of 10 tasks consists of a random (but fixed) pixel-permutation of MNIST and a **fully connected ReLU network** is used. The **task-incremental Split CIFAR 10/100** [13] consists of six 10-way classification tasks, the first is CIFAR 10, and the other ones are extracted from CIFAR 100. The **keras default** CIFAR 10 **convolutional architecture** (with dropout and max-pooling) is used [66].

The only difference to the setup [13] is that like [67], we usually re-initialise network weights after each task, observing better performance. Concretely, we run each method with and without re-initialisation (tuning the regularisation strength independently) and report the better results. On MNIST the improvements with this trick typically below 0.5% and on CIFAR vary between 0% and around 2%.

A known limitation of regularisation methods is that they are not applicable to class-incremental scenarios [55, 56], and we do not include experiments in this setting.

Code is available on github[4]. For the TinyImageNet experiments in Table 3.2, we follow [63], see also Appendix F.8 of Benzing [54].

3.6 DISCUSSION

We have investigated regularisation approaches for continual learning, which are the method of choice for continual learning without replaying old data or expanding the model. We have provided strong theoretical and experimental evidence that both MAS and SI approximate the square root of the Fisher Information. While the square root of the Fisher has no clear theoretical interpretation itself, our analysis makes explicit how MAS and SI are related to second-order information contained in the Fisher. This provides a more plausible explanation of the effectiveness for MAS- and SI based algorithms. In addition, it shows how the three main regularisation methods are related to the same theoretically justified quantity, providing a unified view of these algorithms and their follow ups.

Moreover, our algorithm SIU to approximate SI's path integral can be used for the (non continual learning) algorithm LCA [58], which relies on an expensive approximation of the same integral. This opens up new opportunities to apply LCA to larger models and datasets.

For SI, our analysis included uncovering its bias. We found that the bias explains performance better than the path integral motivating SI. Our theory offers a sound, empirically confirmed explanation for this otherwise surprising finding. Beyond this theoretical contribution, by proposing the algorithm SOS we gave a concrete example how understanding the inner workings of SI leads to substantial performance improvements.

4 https://github.com/freedbee/continual_regularisation

3.7 SUPPLEMENTARY MATERIAL

We provide some supplementary discussion and details related to the main results presented above. We will omit many details and refer to Benzing [54] for more complete information.

We start with a tabluar overview of the different algorithms discussed above, see Table 3.3.

Name	**Paramater Importance** $\boldsymbol{\lambda}(\cdot)$
SI	$\sum_t (g_t + \sigma_t)\Delta(t)$
SIU (SI-Unbiased)	$\sum_t (g_t + \sigma'_t)\Delta(t)$
SIB (SI Bias-only)	$\sum_t (\sigma_t - \sigma'_t)\Delta(t)$
SOS (simple)	$\frac{1-\beta_2}{1-\beta_2^{T+1}} \sum_{t \leq T} \beta_2^{T-t}(g_t + \sigma_t)^2$
SOS (2048, unbiased)	$\frac{1-\beta_2}{1-\beta_2^{T+1}} \sum_{t \leq T} \beta_2^{T-t}\big((g_t + \sigma_t) - \alpha(g_t + \sigma'_t)\big)^2$
Fisher (EWC)	$\frac{1}{N}\sum_X \mathbb{E}_{y \sim \mathbf{p}_X}\left[g(X,y)^2\right]$
AF	$\frac{1}{N}\sum_X \mathbb{E}_{y \sim \mathbf{p}_X}\left[\lvert g(X,y)\rvert\right]$
MAS	$\frac{1}{N}\sum_X \left\lvert \frac{\partial \lvert\lvert \mathbf{p}_X \rvert\rvert^2}{\partial \mathbf{w}} \right\rvert$

TABLE 3.3: **Summary of Regularisation Methods and Related Baselines.**
Notation and Details: Algorithms on the top calculate importance 'online' along the parameter trajectory during training. Algorithms on the bottom calculate importance at the end of training a task by going through (part of) the training set again. Thus, the sum is over timesteps t (top) or datapoints X (bottom). N is the number of images over which is summed. For a datapoint X, $\mathbf{q}_X$ denotes the predicted label distribution and $g(X,y)$ refers to the gradient of the negative log-likelihood of (X,y). $\Delta(t) = \theta(t+1) - \theta(t)$ refers to the parameter update at time t, which depends on both the current task's loss and the auxiliary regularisation loss. Moreover, $(g_t + \sigma_t)$ refers to the stochastic gradient estimate of the current task's loss (where g_t is the full gradient and σ_t the noise) given to the optimizer to update parameters. In contrast, $(g_t + \sigma'_t)$ refers to an independent stochastic gradient estimate.
Algorithms SI, SIU, SIB rescale their final importances as in equation (3.6) for fair comparison. For description, justification and choice of α in SOS, see Supplementary 3.7.3.

3.7.1 *Potential Advantages of SOS vs SI*

Here, we discuss some problems that SI may have and for which SOS offers a remedy due to its more principled nature. We also explain how the points discussed here could explain findings of a large scale empirical comparison of regularisation methods [63] and leave testing these hypotheses to future work. We note that it seems difficult/impossible to even generate hypothesis for some of the findings in [63] without our contribution. Further, if these hypotheses are true, the described shortcomings are addressed by SOS.

1. **Trainingset Size and Difficulty:** Recall that SI is similar to an unnormalised decaying average of $(g + \sigma)^2$. Since the average is unnormalised, the constant of proportionality between $\boldsymbol{\lambda}(\text{SI})$ and $\sqrt{v_t}$ will depend: (1) On the number of summands in the importance of SI, i.e. the number of training iterations and (2) On the speed of gradient decay, which may well depend on the training set difficulty. Thus, SI may underestimate the importance of small datasets (for which fewer updates are performed if the number of epochs is kept constant as e.g. in [13, 63]) and also for easy datasets with fast gradient decay.
   How would we expect this to affect performance? If easy datasets are presented first, then SI will undervalue their importance and thus forget them later on. If hard tasks are presented first, SI will overvalue their importance and make the network less plastic – as latter tasks are easy, this is not too big of a problem. So if easy tasks are presented first, we expect SI to perform worse on average and forget more than when hard tasks are presented first. Strikingly, both lower average accuracy as well as higher forgetting are exactly what [63] (Table 4) observe for SI when easy tasks are presented first (but not for other methods).

2. **Learning Rate Decay:** In our derivation in Section 3.3.2 we assumed a constant learning rate $\eta_t = \eta_0$. If, however, learning rate decay is used, then this will counteract the importance of SI being a decaying average of $(g + \sigma)^2$; for a deacying leanring rate more weight will be put on earlier gradients, which will likely make the estimate of the Fisher less similar to the Fisher at the end of training. This may explain why SI sometimes has worse performance than EWC, MAS in experiments of [63]

3. **Choice of Optimizer:** Note that the bias and thus importance of SI depend on the optimizer. For example if we use SGD with learning

rate η_t (with or without momentum), following the same calculations as before shows that $\boldsymbol{\lambda}(\mathrm{SI}) \approx \sum_t \eta_t (g_t + \sigma_t)^2$. On the one hand, this is related to the Fisher so it may give good results, on the other hand it may overvalue gradients early in training, especially when combined with learning rate decay. [63] use SGD with learning rate decay, which again may be why SI sometimes performs worse than competitors. In an extreme case, when we approximate natural gradient descent by preconditioning with the squared gradient, the SI importance will be constant across parameters, showcasing another, probably undesired, dependence of SI on the choice of optimizer.

4. **Effect of Regularisation:** We already discussed in the main part how strong regularisation influences SI's importance measure. In fact, on CIFAR tasks 2-6 many weights have negative importances, since the regularistation gradient points in the opposite direction of the task gradients and is larger, compare e.g. Figure 3.1. Negative importances seem counterproductive in any theoretical framework.

5. **Batch Size:** We already predicted and confirmed that large batchsizes hurt SI and proposed a remedy to this issue, see also Supplementary 3.7.3.

3.7.2 *Related Work*

The problem of catastrophic forgetting in neural networks has been studied for many decades [9, 68, 69]. In the context of deep learning, it received more attention again [10, 70].

We now review the broad body of continual learning algorithms. Following [71], they are often categorised into regularisation-, replay- and architectural approaches.

Regularisation methods have been reviewed in the main part of this chapter above. We note that [59] is often also called a regularisation method, while being conceptually different from the ones described previously.

Replay methods refer to algorithms which either store a small sample or generate data of old distributions and use this data while training on new methods [12, 72–74]. These approaches can – but certaintly do not have to – be seen as investigating how far standard i.i.d.-training can be relaxed towards the (highly non-i.i.d.) continual learning setting without losing too much performance. They are interesting, but usually circumvent the original motivation of continual learning to maintain knowledge *without*

accessing old distributions. Intriguingly, the most effective way to use old data appears to be simply replaying it, i.e. mimicking training with i.i.d. batches sampled from all tasks simultaneously [75].

Architectural methods extend the network as new tasks arrive [19, 76–79]. This can be seen as a study of how old parts of the network can be effectively used to solve new tasks and touches upon transfer learning. Typically, it avoids the challenge of integrating new knowledge into an existing networks. Finally, [55, 56, 80] point out that different continual learning scenarios and assumptions with varying difficulty were used across the literature.[5]

3.7.3 *Second Moment of Gradients, Hessian, Fisher and SOS*

As pointed out in the main part of this chapter, the second moment v_t of the gradient is a common approximation of the Hessian. Here, we briefly review why this is the case, why the approximation becomes worse for large batch sizes and show how to get a better estimate. We note that the relations between Fisher, Hessian, and squared gradients, which are recapitulated here, are discussed in several places in the literature, e.g. [45, 46, 81].

One way to relate the squared gradients to the Hessian is through the Fisher[6]: The Fisher Information is an approximation of the Hessian, which becomes exact when the learned label distribution coincides with the real label distribution. The Fisher takes an expectation over the model's label distribution. A common approximation is to replace this expectation by the (deterministic) labels of the dataset. This is called the *Empirical Fisher* and it is a good approximation of the FIsher if the model classifies most (training) images correctly and confidently. To summarise, the Fisher is a good approximation of the Hessian under the assumption that the model's predicted distribution coincides with the real distribution and - under the same assumption - the empirical Fisher is a good approximation of the Hessian. In fact, taking a closer look at the derivations, it seems plausible that the empirical Fisher is a better approximation of the Hessian than the real Fisher, since - like the Hessian - it uses the real label distribution rather than the model's label distribution.

5 We critically review and question some of their experimental results below.

6 Alternatively, one can directly apply the derivation, which is used to relate Fisher and Hessian, to the squared gradient (Empirical Fisher).

To avoid confusion and as pointed out by [45], we also note that the Empirical Fisher is not generally equal to the Generalised Gauss Newton matrix.

Note that the connection between squared gradients and the Hessian assumes that gradients are squares gradients before averaging. Here, and in many other places e.g. [52, 62], this is approximated by squaring after averaging over a mini-batch since this is easier to implement and requires less computation. We describe below why this approximation is valid for small batch sizes and introduce an improved and easy to compute estimate for large batch sizes. We have not seen this improved estimate elsewhere in the literature.

For this subsection, let us slightly change notation and denote the images by $X_1, \dots, X_D$ and the gradients (with respect to their labels and the cross entropy loss) by $g + \sigma_1, \dots g + \sigma_D$. Here, again g is the overall training set gradient and σ_i is the noise (i.e. $\sum_{i=1}^{D} \sigma_i = 0$) of individual images (rather than mini-batches). Then the Empirical Fisher is given by

$$\mathrm{EF} = \frac{1}{D} \sum_{i=1}^{D} (g + \sigma_i)^2 = g^2 + \mathbb{E}[\sigma_k^2],$$

where k is uniformly drawn from $\{1, \dots, D\}$

Second Moment Estimate of Fisher. We want to compare EF to evaluating the squared gradient of a minibatch. Let $i_1, \dots, i_b$ denote uniformly random, independent indices from $\{1, \dots, D\}$, so that $X_{i_1}, \dots, X_{i_b}$ is a random minibatch of size b drawn with replacement. Let $g + \sigma$ be the gradient on this mini-batch. We then have, taking expectations over the random indices,

$$\begin{aligned}
\mathbb{E}[(g+\sigma)^2] &= \mathbb{E}\left[\frac{1}{b^2} \sum_{r,s=1}^{b} (g + \sigma_{i_r})(g + \sigma_{i_s})\right] \\
&= \frac{b(b-1)}{b^2} \mathbb{E}[(g + \sigma_{i_1})(g + \sigma_{i_2})] + \frac{b}{b^2} \mathbb{E}[(g + \sigma_{i_1})^2] \\
&= \frac{b-1}{b} g^2 + \frac{1}{b} \mathrm{EF} \\
&\approx \frac{1}{b} \mathrm{EF}
\end{aligned}$$

The last approximation is biased, but it is still a decent approximation as long as $\mathbb{E}[\sigma_k^2] \gg b g^2$. This explains why the second moment estimate of the Fisher gets worse for large batch sizes. Note that here σ_k refers to the noise

of the gradient with batch size 1 (whereas σ is the noise of a mini-batch, which is b times smaller).
For an analysis assuming that the mini-batch is drawn without replacement, see e.g. [62]. Note that for a minibatch size of 2048 and a trainingset size of 60000, the difference between drawing with or without replacement is small, as on average there are only 35 duplicates in a batch with replacement, i.e. less than 2% of the batch.

3.7.3.1 *Improved Estimate of Fisher for SOS.*

We use the same notation as above, in particular σ_k refers to the gradient noise of a single randomly sampled image (not an entire minibatch). Let us denote by $g + \sigma$ and $g + \sigma'$ the gradient estimates obtained from two independent minibatches of size b (each sampled with replacement as above). Then for any $\alpha \in \mathbb{R}$, following the same calculations as above and slightly rearranging gives

$$\mathbb{E}\left[\left((g+\sigma) - \alpha(g+\sigma')\right)^2\right] \quad = \quad (1-\alpha)^2 g^2 + \frac{1+\alpha^2}{b}\mathbb{E}[\sigma_k^2],$$

where we used that each minibatch element is drawn independently and that $\mathbb{E}[\sigma_k] = 0$. Now, if $(1-\alpha)^2 = \frac{1+\alpha^2}{b}$, then the above expression is proportional to the Empirical Fisher $\mathrm{EF} = g^2 + \mathbb{E}[\sigma_k^2]$. Since we rescale the regularisation loss with a hyperparameter, proportionality is all we need to get a strictly better (i.e. unbiased) approximation of the Empirical Fisher.

The above condition for α and b is satisfied when $\alpha = \frac{b+\sqrt{2b-1}}{b-1}$. In practice, we used $\alpha = 0$ for the experiments with batchsize 256 and $\alpha = 1$ for the experiments with batchsize 2048. Note that $\alpha = 1$ is always a good approximation when $\mathbb{E}[\sigma_k^2] \gg g^2$ (which we've seen to be the case). In our implementation SOS required calculating gradients on two minibatches, thus doubling the number of forward and backward passes during training. However, one could simply split the existing batch in two halves to keep the number of forward and backward passes constant. For large batch experiments we found introducing α necessary to obtain as good performance as with smaller batch sizes. This also supports our claim that SI's and SOS's importance measures rely on second order information contained in the Fisher.

3.7.4 *Gradient Noise*

Here, we quantitatively assess the noise magnitude outside the continual learning context. Recall that Figure 3.1 (left) already show that the noise dominates the SI importance measure, which indicates that the noise is considerably larger than the gradient itself.

To obtain an assessment independent of the SI continual learning importance measure, we trained our network on MNIST as described before, i.e. a ReLu network with 2 hidden layers of 2000 units each, trained for 20 epochs with batch size 256 and default Adam settings. At each training iteration, on top of calculating the stochastic mini-batch gradient used for optimization, we also computed the full gradient on the entire training set and computed the noise – which refers to the squared ℓ_2 distance between the stochastic mini-batch gradient and the full gradient – as well as the ratio between noise and gradient, measured as the ratio of squared ℓ_2 norms. The results are shown in Figure 3.3 (top). In addition, we computed the fraction of iterations in which the ratio between noise and squared gradient norm is above a certain threshold, see Figure 3.3 (bottom).

3.7.5 *Exact Computation of Bias of SI with Adam*

We claimed that the difference between SI and SIU (green and blue line) seen in Figure 3.1 is due to the term $(1-\beta_1)\sigma_t^2$. To see this, recall that for SI, we approximate $\frac{\partial L(t)}{\partial \mathbf{w}}$ by $g_t + \sigma_t$, which is the same gradient estimate given to Adam. So we get

$$\textbf{SI:} \qquad \frac{\partial L(t)}{\partial \mathbf{w}}\Delta(t) = \frac{(1-\beta_1)(g_t+\sigma_t)^2}{\sqrt{v_t}+\epsilon} + \frac{\beta_1(g_t+\sigma_t)m_{t-1}}{\sqrt{v_t}+\epsilon}.$$

For SIU, we use an independent mini-batch estimate $g_t + \sigma_t'$ for $\frac{\partial L(t)}{\partial \mathbf{w}}$ and therefore obtain

$$\textbf{SIU:} \qquad \frac{\partial L(t)}{\partial \mathbf{w}}\Delta(t) = \frac{(1-\beta_1)(g_t+\sigma_t')(g_t+\sigma_t)}{\sqrt{v_t}+\epsilon} + \frac{\beta_1(g_t+\sigma_t')m_{t-1}}{\sqrt{v_t}+\epsilon}.$$

Taking the difference between these two and ignoring all terms which have expectation zero (note that $\mathbb{E}[\sigma_t] = \mathbb{E}[\sigma_t'] = 0$ and that σ_t, σ_t' are independent of m_{t-1} and g_t) gives

$$\textbf{SI} - \textbf{SIU:} \qquad (1-\beta_1)\frac{\sigma_t^2}{\sqrt{v_t}+\epsilon}$$

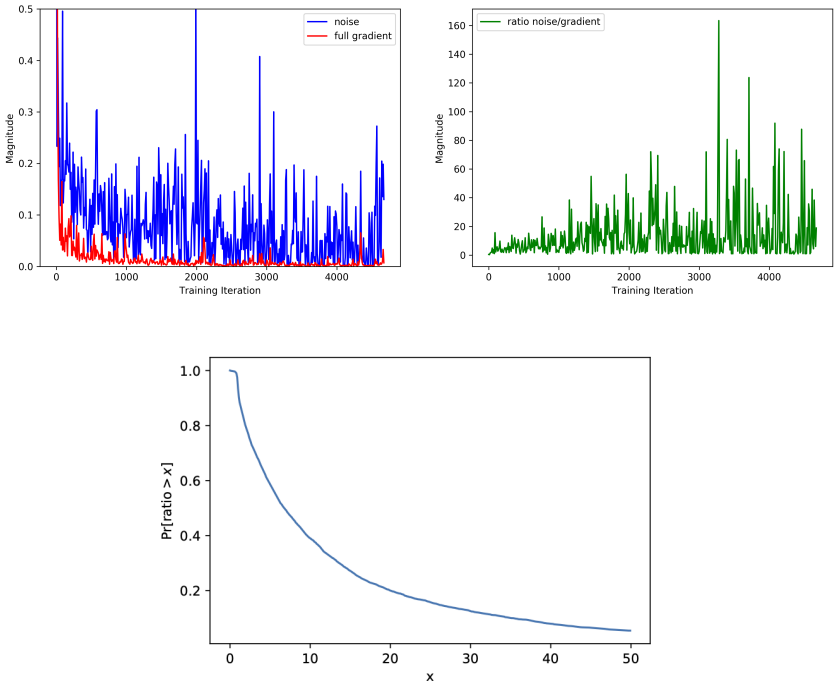


FIGURE 3.3: **Top:** Gradient noise, measured as squared ℓ_2 distance between full training set gradient and the stochastic mini-batch gradient with a batch size of 256. 'Full gradient' magnitude is also measured as squared ℓ_2 norm.
Data obtained by training a ReLu network with 2 hidden layers of 2000 hidden units for 20 epochs with default Adam settings. Only every 20-th datapoint shown for better visualisation.
Bottom: Same data. y-value shows fraction of training iterations in which the ratio between mini-batch noise and full training set gradient was at least x-value. In particular the batch-size was 256. 'Ratio' refers to the ratio of squared ℓ_2 norms of the respective values.

as claimed.

Note also that in expectation SIU equals $(1-\beta_1)\frac{g_t^2}{\sqrt{v_t}+\epsilon} + \beta_1 \frac{g_t m_{t-1}}{\sqrt{v_t}+\epsilon}$ so that a large difference between SI and SIU really means $(1-\beta_1)\sigma_t^2 \gg (1-\beta_1)g_t^2 + \beta_1 m_{t-1} g_t \approx \beta_1 m_{t-1} g_t$. The last approximation here is valid because $\beta_1 m_{t-1} \gg (1-\beta_1)g_t$ which holds since (1) $\beta_1 \gg (1-\beta_1)$ and (2) $\mathbb{E}[|m_{t-1}|] \geq g_t$ since $\mathbb{E}[|m_{t-1}|] \geq \mathbb{E}[m_{t-1}] \approx \mathbb{E}[|g_t|]$.

4

PRESYNAPTIC STOCHASTICITY IMPROVES ENGERY-EFFICIENCY AND HELPS ALLEVIATE STABILITY-PLASTICITY DILEMMA

This chapter is mostly taken and partially adapted from Schug, Benzing & Steger [82], which appeared in eLife in 2021 and is joint work with Simon Schug and Angelika Steger. Simon Schug and Frederik Benzing contributed equally and Frederik Benzing was main responsible for the theoretical justification of the learning rule.

4.1 CONTEXT

This chapter applies some of the insights gained in Chapter 3 in the context of biological neural networks, but also has additional contributions. Here, to provide context for several other parts of the motivation for this study, we present it within the context of neuroscience.

4.2 INTRODUCTION

It has long been known that synaptic signal transmission is stochastic [83]. When an action potential arrives at the presynapse, there is a high probability that no neurotransmitter is released – a phenomenon observed across species and brain regions [84]. From a computational perspective, synaptic stochasticity seems to place unnecessary burdens on information processing. Large amounts of noise hinder reliable and efficient computation [85, 86] and synaptic failures appear to contradict the fundamental evolutionary principle of energy-efficient processing [87]. The brain, and specifically action potential propagation consume a disproportionately large fraction of energy [88, 89] – so why propagate action potentials all the way to the synapse only to ignore the incoming signal there?

To answer this neurocomputational enigma various theories have been put forward, see Llera-Montero, Sacramento & Costa [90] for a review. One important line of work proposes that individual synapses do not merely maximise information transmission, but rather take into account metabolic costs, maximising the information transmitted *per unit of energy* [91]. This approach has proven fruitful to explain synaptic failures [89, 92],

low average firing rates [91] as well as excitation-inhibition balance [93] and is supported by fascinating experimental evidence suggesting that both presynaptic glutamate release [94] and postsynaptic channel properties [95, 96] are tuned to maximise information transmission per energy.

However, so far information-theoretic approaches have been limited to signal transmission at single synapses, ignoring the context and goals in which the larger network operates. As soon as context and goals guide network computation certain pieces of information become more relevant than others. For instance, when reading a news article the textual information is more important than the colourful ad blinking next to it – even when the latter contains more information in a purely information-theoretic sense.

Here, we study presynaptic stochasticity on the network level rather than on the level of single synapses. We investigate its effect on (1) energy efficiency and (2) the stability-plasticity dilemma in model neural networks that learn to selectively extract information from complex inputs.

We find that presynaptic stochasticity in combination with presynaptic plasticity allows networks to extract information at lower metabolic cost by sparsely allocating energy to synapses that are important for processing the given stimulus. As a result, presynaptic release probabilities encode synaptic importance. We show that this notion of importance is related to the Fisher Information, a theoretical measure for the network's sensitivity to synaptic changes.

Building on this finding and previous work [11] we explore a potential role of presynaptic stochasticity in the stability-plasticity dilemma. In line with experimental evidence [97, 98], we demonstrate that selectively stabilising important synapses improves lifelong learning. Furthermore, these experiments link presynaptically induced sparsity to improved memory.

4.3 MODEL

Our goal is to understand how information processing and energy consumption are affected by stochasticity in synaptic signal transmission. While there are various sources of stochasticity in synapses, here, we focus on modelling *synaptic failures* where action potentials at the presynapse fail to trigger any postsynaptic depolarisation. The probability of such failures is substantial [84, 99, 100] and, arguably, due to its all-or-nothing-characteristic has the largest effect on both energy consumption and information transmission.

As a growing body of literature suggests, Artifical Neural Networks (ANNs) match several aspects of biological neuronal networks in various

goal-driven situations [101–106]. Crucially, they are the only known model to solve complex vision and reinforcement learning tasks comparably well as humans. We therefore choose to extend this class of models by explicitly incorporating synaptic failures and study their properties in a number of complex visual tasks.

4.3.1 *Model Details*

The basic building blocks of ANNs are neurons that combine their inputs $a_1, \ldots, a_n$ through a weighted sum $w_1 a_1 + \ldots w_n a_n$ and apply a nonlinear activation function $\sigma(\cdot)$. The weights w_i naturally correspond to *synaptic strengths* between presynaptic neuron i and the postsynaptic neuron. Although synaptic transmission is classically described as a binomial process [83] most previous modelling studies assume the synaptic strengths to be deterministic. This neglects a key characteristic of synaptic transmission: the possibility of synaptic failures where no communication between pre- and postsynapse occurs at all.

In the present study, we explicitly model presynaptic stochasticity by introducing a random variable $r_i \sim \text{Bernoulli}(p_i)$, whose outcome corresponds to whether or not neurotransmitter is released. Formally, each synapse w_i is activated stochastically according to

$$w_i = r_i \cdot m_i, \qquad \text{where } r_i \sim \text{Bernoulli}(p_i) \tag{4.1}$$

so that it has expected synaptic strength $\bar{w}_i = p_i m_i$. The postsynaptic neuron calculates a stochastic weighted sum of its inputs with a nonlinear activation

$$a^{\text{post}} = \sigma\left(\sum_{i=1}^{n} w_i a^{\text{pre}}\right). \tag{4.2}$$

During learning, synapses are updated and both synaptic strength and release probability are changed. We resort to standard learning rules to change the expected synaptic strength. For the multilayer perceptron, this update is based on stochastic gradient descent with respect to a loss function $L(\bar{w}, p)$, which in our case is the standard cross-entropy loss. Concretely, we have

$$\bar{w}_i^{(t+1)} = \bar{w}_i^{(t)} - \eta g_i, \qquad \text{where } g_i = \frac{\partial L(\bar{w}^{(t)}, p)}{\partial \bar{w}_i^{(t)}} \tag{4.3}$$

where the superscript corresponds to time steps. Note that this update is applied to the expected synaptic strength $\bar{w}_i$, requiring communication between pre- and postsynape, see also Discussion. For the explicit update rule of the synaptic strength m_i see Materials and Methods, equation (4.8). For the standard perceptron model, g_i is given by its standard learning rule [107]. Based on the intuition that synapses which receive larger updates are more important for solving a given task, we update p_i using the update direction g_i according to the following simple scheme

$$p_i^{(t+1)} = \begin{cases} p_i^{(t)} + p_{\text{up}}, & \text{if } |g_i| > g_{\text{lim}}, \\ p_i^{(t)} - p_{\text{down}}, & \text{if } |g_i| \leq g_{\text{lim}}. \end{cases} \tag{4.4}$$

Here, $p_{\text{up}}, p_{\text{down}}, g_{\text{lim}}$ are three metaplasticity parameters shared between all synapses.[1] To prevent overfitting and to test robustness, we tune them using one learning scenario and keep them fixed for all other scenarios, see Materials and Methods. To avoid inactivated synapses with release probability $p_i = 0$ we clamp p_i to stay above 0.25, which we also use as the initial value of p_i before learning.

On top of the above intuitive motivation, we give a theoretical justification for this learning rule in Materials and Methods, showing that synapses with larger Fisher Information obtain high release probabilities, also see Figure 4.2d.

4.3.2 *Measuring Energy Consumption*

For our experiments, we would like to quantify the energy consumption of the neural network. Harris, Jolivet & Attwell [89] find that the main constituent of neural energy demand is synaptic signal transmission and that the cost of synaptic signal transmission is dominated by the energy needed to reverse postsynaptic ion fluxes. In our model, the component most closely matching the size of the postsynaptic current is the expected synaptic strength, which we therefore take as measure for the model's energy consumption. In the Supplementary, we also measure the metabolic cost incurred by the activity of neurons by calculating their average rate of activity.

1 We point out that in a noisy learning setting the gradient g does not decay to 0, so that the learning rule in (4.4) will maintain network function by keeping certain release probabilities high. See also Material and Methods for a theoretical analysis.

Box 4.1: Mutual Information

The Mutual Information $I(Y;Z)$ of two jointly distributed random variables Y, Z is a common measure of their dependence [85]. Intuitively, mutual information captures how much information about Y can be obtained from Z, or vice versa. Formally, it is defined as

$$I(Y;Z) \equiv H(Y) - H(Y|Z) = H(Z) - H(Z|Y)$$

where $H(Y)$ is the entropy of Y and $H(Y|Z)$ is the conditional entropy of Y given Z.
In our case, we want to measure how much task-relevant information Y is contained in the neural network output Z. For example, the neural network might receive as input a picture of a digit with the goal of predicting the identity of the digit. Both the ground-truth digit identity Y and the network's prediction Z are random variables depending on the random image X. The measure $I(Y;Z)$ quantifies how much of the behaviourally relevant information Y is contained in the network's prediction Z ignoring irrelevant information also present in the complex, high-entropy image X.

4.3.3 *Measuring Information Transmission*

We would like to measure how well the neural network transmits information relevant to its behavioural goal. In particular, we are interested in the setting where the complexity of the stimulus is high relative to the amount of information that is relevant for the behavioural goal. To this end, we present complex visual inputs with high information content to the network and teach it to recognise the object present in the image. We then measure the mutual information between network output and object identity, see Box 4.1.

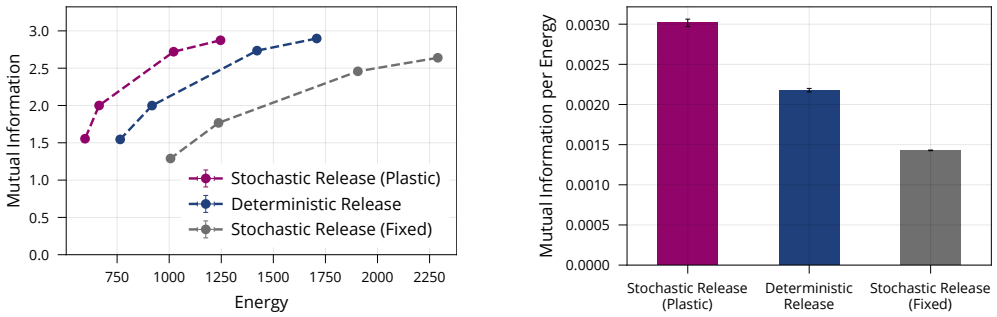
4.4 RESULTS

4.4.1 *Presynaptic Stochasticity Enables Energy-Efficient Information Processing*

We now investigate the energy efficiency of a network that learns to classify digits from the MNIST handwritten digit dataset [108]. The inputs are high-dimensional with high entropy, but the relevant information is simply the identity of the digit. We compare the model with plastic, stochastic release to two controls. A standard ANN with deterministic synapses is included to investigate the combined effect of presynaptic stochasticity and plasticity. In addition, to isolate the effect of presynaptic plasticity, we introduce a control which has stochastic release, but with a fixed probability. In this control, the release probability is identical across synapses and chosen to match the average release probability of the model with plastic release after it has learned the task.

All models are encouraged to find low-energy solutions by penalising large synaptic weights through standard ℓ_2-regularisation. Figure 4.1a shows that different magnitudes of ℓ_2-regularisation induce different information-energy trade-offs for all models, and that the model with plastic, stochastic release finds considerably more energy-efficient solutions than both controls, while the model with non-plastic release requires more energy then the deterministic model. Together, this supports the view that a combination of presynaptic stochasticity and plasticity promotes energy-efficient information extraction.

In addition, we investigate how stochastic release helps the network to lower metabolic costs. Intuitively, a natural way to save energy is to assign high release probabilities to synapses that are important to extract relevant information and to keep remaining synapses at a low release probability. Figure 4.2a shows that after learning, there are indeed few synapses with high release probabilities, while most release probabilities are kept low. We confirm that this sparsity develops independently of the initial value of release probabilities before learning, see Supplementary Figure 4.6d. To test whether the synapses with high release probabilities are most relevant for solving the task we perform a lesion experiment. We successively remove synapses with low release probability and measure how well the lesioned network still solves the given task, see Figure 4.2b. As a control, we remove synapses in a random order independent of their release probability. We find that maintaining synapses with high release probabilities is significantly more important to network function than maintaining random ones.



(a) (b)

FIGURE 4.1: **Energy Efficiency of Model with Stochastic and Plastic Release.** **(a)** Different trade-offs between mutual information and energy are achievable in all network models. Generally, stochastic synapses with learned release probabilities are more energy-efficient than deterministic synapses or stochastic synapses with fixed release probability. The fixed release probabilities model was chosen to have the same average release probability as the model with learned probabilities. **(b)** Best achievable ratio of information per energy for the three models from (a). Error bars in (a) and (b) denote the standard error for three repetitions of the experiment.

Moreover, we find, as expected, that synapses with high release probabilities consume considerably more energy than synapses with low release probability, see Figure 4.2c. This supports the hypothesis that the model identifies important synapses for the task at hand and spends more energy on these synapses while saving energy on irrelevant ones.

We have seen that the network relies on a sparse subset of synapses to solve the task efficiently. However, sparsity is usually thought of on a neuronal level, with few neurons rather than few synapses encoding a given stimulus. Therefore, we quantify sparsity of our model on a neuronal level. For each neuron we count the number of 'important' input- and output synapses, where we define 'important' to correspond to a release probability of at least $p = 0.9$. Note that the findings are robust with respect to the precise value of p, see Figure 4.2a. We find that the distribution of important synapses per neuron is inhomogeneous and significantly different from a randomly shuffled baseline with a uniform distribution of active synapses (Kolmogorov-Smirnoff test, $D = 0.505, p < 0.01$), see Figure 4.3a. Thus, some neurons have disproportionately many important inputs, while others have very few, suggesting sparsity on a neuronal level. As additional quantification of this effect, we count the number of highly

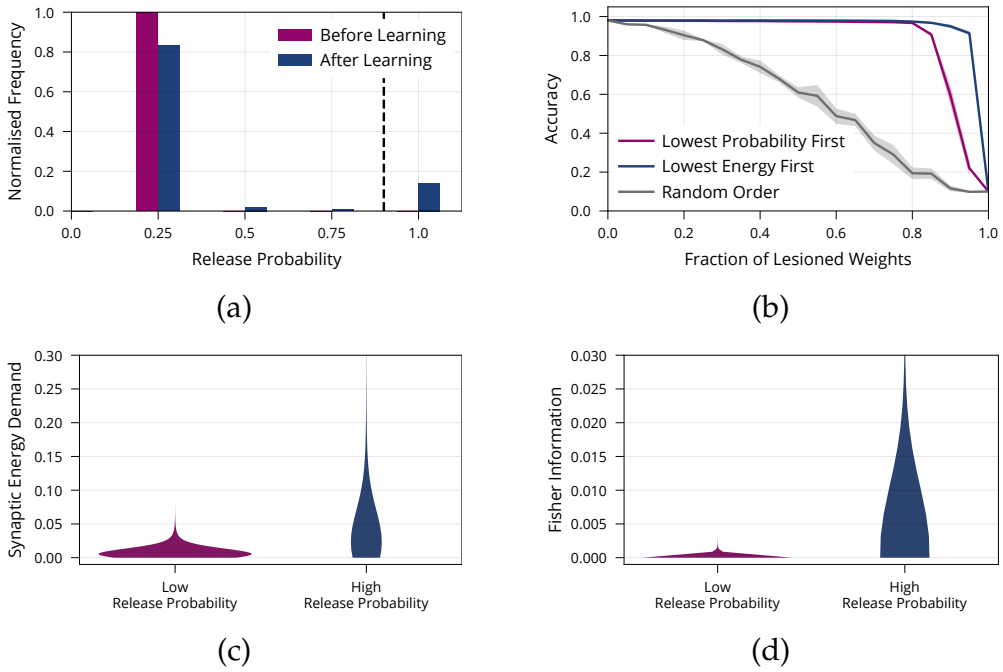


FIGURE 4.2: **Importance of Synapses with High Release Probability for Network Function. (a)** Histogram of release probabilities before and after learning, showing that the network relies on a sparse subset of synapses to find an energy-efficient solution. Dashed line at $p = 0.9$ indicates our boundary for defining a release probability as 'low' or 'high'. We confirmed that results are independent of initial value of release probabilities before learning (see Supplementary, Figure 4.6d). **(b)** Accuracy after performing the lesion experiment either removing synapses with low release probabilities first or removing weights randomly, suggesting that synapses with high release probability are most important for solving the task. **(c)** Distribution of synaptic energy demand for high and low release probability synapses. **(d)** Distribution of the Fisher information for high and low release probability synapses. It confirms the theoretical prediction that high release probability corresponds to high Fisher Information. All panels show accumulated data for three repetitions of the experiment. Shaded regions in (b) show standard error.

important neurons, where we define a neuron to be highly important if its number of active inputs is two standard deviations below or above the mean (mean and standard deviation from shuffled baseline). We find that our model network with presynaptic stochasticity has disproportionate numbers of highly important and unimportant neurons, see Figure 4.3b. Moreover, we check whether neurons with many important inputs tend to

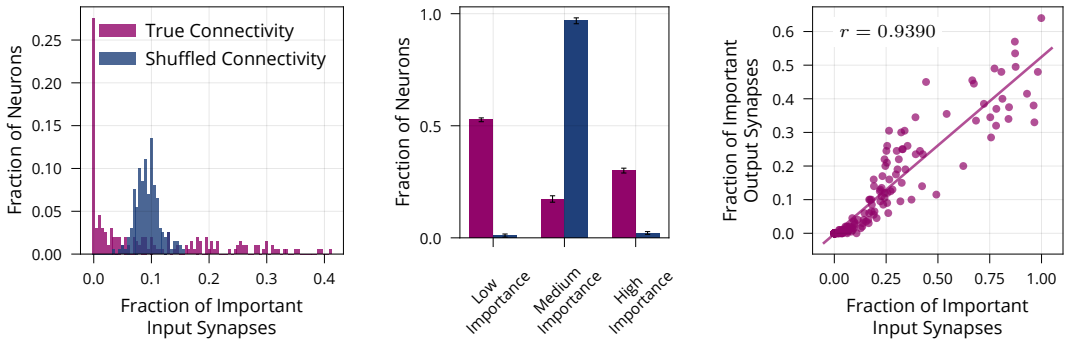


FIGURE 4.3: **Neuron-Level Sparsity of Network after Learning. (a)** Histogram of the fraction of important input synapses per neuron for second layer neurons after learning for true and randomly shuffled connectivity (see Supplementary, Figure 4.7a for other layers). **(b)** Same data as (a), showing number of low/medium/high importance neurons, where high/low importance neurons have at least two standard deviations more/less important inputs than the mean of random connectivity. **(c)** Scatter plot of first layer neurons showing the number of important input and output synapses after learning on MNIST, Pearson correlation is $r = 0.9390$ (see Supplementary, Figure 4.7b for other layers). Data in (a) and (c) are from one representative run, error bars in (b) show standard error over three repetitions.

have many important outputs, indeed finding a correlation of $r = 0.93$, see Figure 4.3c. These analyses all support the claim that the network is sparse not only on a synaptic but also on a neuronal level.

Finally, we investigate how release probabilities evolve from a theoretical viewpoint under the proposed learning rule. Note that the evolution of release probabilities is a random process, since it depends on the random input to the network. Under mild assumptions, we show (Materials and Methods) that release probabilities are more likely to increase for synapses with large Fisher Information[2]. Thus, synapses with large release probabilities will tend to have high Fisher Information. We validate this theoretical prediction empirically, see Figure 4.2d.

2 In this context, the Fisher Information is a measure of sensitivity of the network to changes in synapses, measuring how important preserving a given synapse is for network function.

4.4.2 *Presynaptically Driven Consolidation Helps Alleviate the Stability-Plasticity Dilemma*

While the biological mechanisms addressing the stability-plasticity dilemma are diverse and not fully understood, it has been demonstrated experimentally that preserving memories requires maintaining the synapses which encode these memories [97, 98, 109]. In this context, theoretical work suggests that the Fisher Information is a useful way to quantify which synapses should be maintained [11]. Inspired by these insights, we hypothesise that the synaptic importance encoded in release probabilities can be used to improve the network's memory retention by selectively stabilising important synapses.

We formalise this hypothesis in our model by lowering the learning rate (plasticity) of synapses according to their importance (release probability). Concretely, the learning rate $\eta = \eta(p_i)$ used in (4.3) is scaled as follows

$$\eta(p_i) = \eta_0 \cdot (1 - p_i). \tag{4.5}$$

such that the learning rate is smallest for important synapses with high release probability. η_0 denotes a base learning rate that is shared by all synapses. We complement this consolidation mechanism by freezing the presynaptic release probabilities p_i once they have surpassed a predefined threshold p_{freeze}. This ensures that a synapse whose presynaptic release probability was high for a previous task retains its release probability even when unused during consecutive tasks. In other words, the effects of presynaptic LTD are assumed to act on a slower timescale than learning single tasks. Note that the freezing mechanism ensures that all synaptic strengths $\bar{w}_i$ retain a small degree of plasticity, since the learning rate modulation factor $(1 - p_i)$ remains greater than 0.

To test our hypothesis that presynaptically driven consolidation allows the network to make improved stability-plasticity trade-offs, we sequentially present a number of tasks and investigate the networks behaviour. We mainly focus our analysis on a variation of the MNIST handwritten digit dataset, in which the network has to successively learn the parity of pairs of digits, see Figure 4.4a. Additional experiments are reported in the Supplementary Material, see Table 4.1.

First, we investigate whether presynaptic consolidation improves the model's ability to remember old tasks. To this end, we track the accuracy on the first task over the course of learning, see Figure 4.4b. As a control, we include a model without consolidation and with deterministic synapses.

While both models learn the first task, the model without consolidation forgets more quickly, suggesting that the presynaptic consolidation mechanism does indeed improve memory.

Next, we ask how increased stability interacts with the network's ability to remain plastic and learn new tasks. To assess the overall trade-off between stability and plasticity we report the average accuracy over all five tasks, see Figure 4.4c.

We find that the presynaptic consolidation model performs better than a standard model with deterministic synapses and without consolidation. In addition, we compare performance to two state-of-the art machine learning algorithms: The well-known algorithm EWC [11] explicitly relies on the Fisher Information and performs a separate consolidation phase after each task. BGD [110] is a Bayesian approach that models synapses as distributions, but does not capture the discrete nature of synaptic transmission. The presynaptic consolidation mechanism performs better than both these state-of-the-art machine learning algorithms, see Figure 4.4c. Additional experiments in the Supplementary suggest overall similar performance of Presynaptic Consolidation to BGD and similar or better performance than EWC.

To determine which components of our model contribute to its lifelong learning capabilities, we perform an ablation study, see Figure 4.5a. We aim to separate the effect of (1) consolidation mechanisms and (2) presynaptic plasticity.

First, we remove the two consolidation mechanisms, learning rate modulation and freezing release probabilities, from the model with stochastic synapses. This yields a noticeable decrease in performance during lifelong learning, thus supporting the view that stabilising important synapses contributes to addressing the stability-plasticity dilemma.

Second, we aim to disentangle the effect of presynaptic plasticity from the consolidation mechanisms. We therefore introduce a control in which presynaptic plasticity but not consolidation is blocked. Concretely, the control has 'ghost release probabilities' $\tilde{p}_i$ evolving according to equation (4.4) and modulating plasticity according to equation (4.5); but the synaptic release probability is fixed at 0.5. We see that this control performs worse than the original model with a drop in accuracy of 1.4% on Split MNIST ($t = 3.44$, $p < 0.05$) and a drop of accuracy of 5.6% on Permuted MNIST ($t = 6.72, p < 0.01$). This suggests that presynaptic plasticity, on top of consolidation, helps to stabilise the network. We believe that this can be

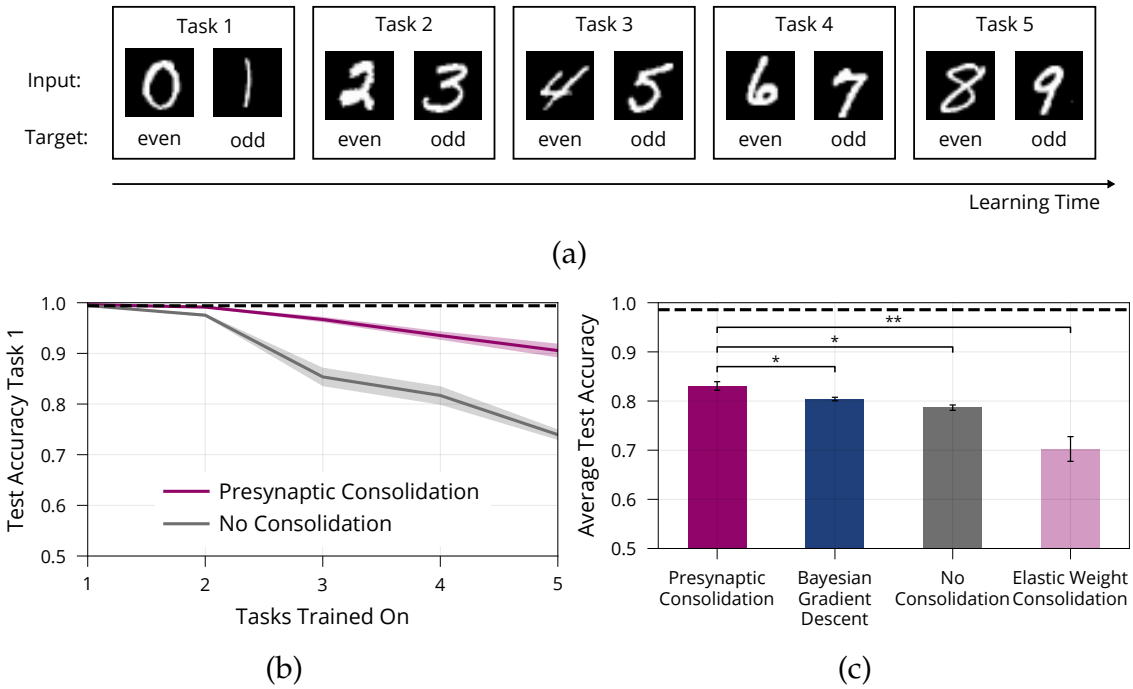


FIGURE 4.4: **Lifelong Learning in a Model with Presynaptically Driven Consolidation. (a)** Schematic of the lifelong learning task Split MNIST. In the first task the model network is presented 0s and 1s, in the second task it is presented 2s and 3s, etc. For each task the model has to classify the inputs as even or odd. At the end of learning, it should be able to correctly classify the parity of all digits, even if a digit has been learned in an early task. **(b)** Accuracy of the first task when learning new tasks. Consolidation leads to improved memory preservation. **(c)** Average accuracies of all learned tasks. The presynaptic consolidation model is compared to a model without consolidation and two state-of-the-art machine learning algorithms. Differences to these models are significant in independent t-tests with either $p < 0.05$ (marked with *) or with $p < 0.01$ (marked with **). Dashed line indicates an upper bound for the network's performance, obtained by training on all tasks simultaneously. Panels (b) and (c) show accumulated data for three repetitions of the experiment. Shaded regions in (b) and error bars in (c) show standard error.

attributed to the sparsity induced by the presynaptic plasticity which decreases overlap between different tasks.

The above experiments rely on a gradient-based learning rule for multilayer perceptrons. To test whether presynaptic consolidation can also alleviate stability-plasticity trade-offs in other settings, we study its effects on learning in a standard perceptron [107]. We train the perceptron sequen-

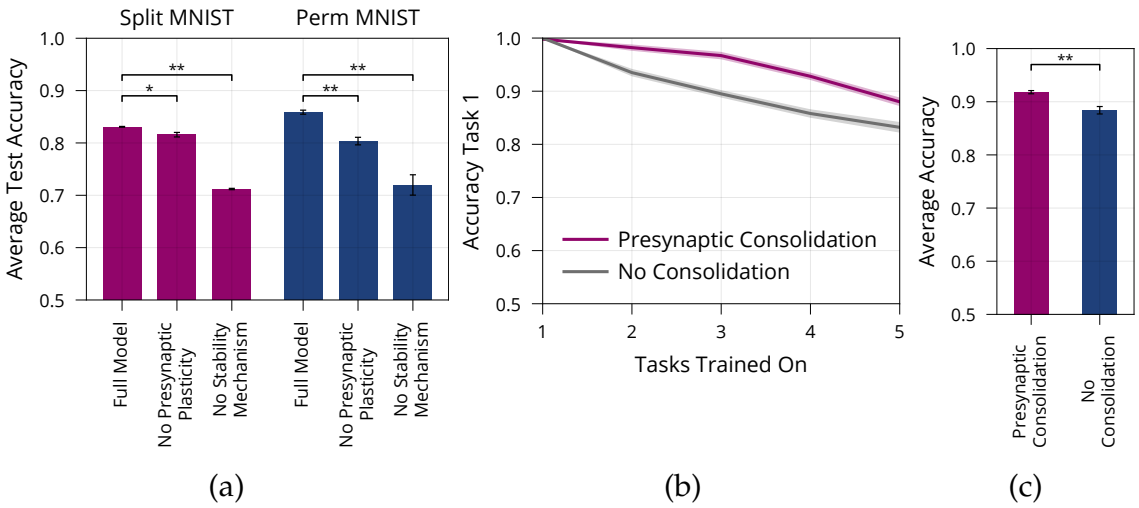


FIGURE 4.5: **Model Ablation and Lifelong Learning in a Standard Perceptron.** **(a)** Ablation of the Presynaptic Consolidation model on two different lifelong learning tasks, see full text for detailed description. Both presynaptic plasticity and synaptic stabilisation significantly improve memory. **(b)+(c)** Lifelong Learning in a Standard Perceptron akin to Figure 4.4b, 4.4c, showing the accuracy of the first task when learning consecutive tasks in (b) as well as the average over all five tasks after learning all tasks in (c). Error bars and shaded regions show standard error of three respectively ten repetitions, in (a) respectively (b+c). All pair-wise comparisons are significant, independent t-tests with $p < 0.01$ (denoted by **) or with $p < 0.05$ (denoted by *).

tially on five pattern memorisation tasks, see Materials and Methods for full details. We find that the presynaptically consolidated perceptron maintains a more stable memory of the first task, see Figure 4.5b. In addition, this leads to an overall improved stability-plasticity trade-off, see Figure 4.5c and shows that the effects of presynaptic consolidation in our model extend beyond gradient-based learning.

4.5 DISCUSSION

4.5.1 *Main Contribution*

Information transmission in synapses is stochastic. While previous work has suggested that stochasticity allows to maximise the amount of information transmitted per unit of energy spent, this analysis has been restricted to single synapses. We argue that the relevant quantity to be considered is

task-dependent information transmitted by entire networks. Introducing a simple model of the all-or-nothing nature of synaptic transmission, we show that presynaptic stochasticity enables networks to allocate energy more efficiently. We find theoretically as well as empirically that learned release probabilities encode the importance of weights for network function according to the Fisher Information. Based on this finding, we suggest a novel computational role for presynaptic stochasticity in lifelong learning. Our experiments provide evidence that coupling information encoded in the release probabilities with modulated plasticity can help alleviate the stability-plasticity dilemma.

4.5.2 *Modelling Assumptions and Biological Plausibility*

4.5.2.1 *Stochastic Synaptic Transmission*

Our model captures the occurrence of synaptic failures by introducing a Bernoulli random variable governing whether or not neurotransmitter is released. Compared to classical models assuming deterministic transmission, this is one step closer to experimentally observed binomial transmission patterns, which are caused by multiple, rather than one, release sites between a given neuron and dendritic branch. Importantly, our simplified model accounts for the event that there is no postsynaptic depolarisation at all. Even in the presence of multiple release sites, this event has non negligible probability: Data from cultured hippocampal neurons [111, Figure 2D] and the neocortex [99, Figure 7C] shows that the probability $(1-p)^N$ that none of N release sites with release probability p is active, is around 0.3-0.4 even for N as large as 10. More recent evidence suggests an even wider range of values depending on the extracellular calcium concentration [100].

4.5.2.2 *Presynaptic Long-Term Plasticity*

A central property of our model builds on the observation that the locus of expression for long-term plasticity can both be presynaptic and postsynaptic [112–117]. The mechanisms to change either are distinct and synapse-specific [118, 119], but how exactly pre- and postsynaptic forms of LTP and LTD interact is not yet fully understood [120]. The induction of long-term plasticity is thought to be triggered postsynaptically for both presynaptic and postsynaptic changes [118, 121] and several forms of presynaptic plasticity are known to require retrograde signalling [120], for example through nitric oxide or endocannabinoids [117, 122, 123]. This interaction between

pre- and postsynaptic sites is reflected by our learning rule, in which both pre- and postsynaptic changes are governed by postsynaptic updates and require communication between pre- and postsynapse. The proposed presynaptic updates rely on both presynaptic LTP and presynaptic LTD. At least one form of presynaptic long-term plasticity is known to be bidirectional switching from potentiation to depression depending on endocannabinoid transients [124, 125].

4.5.2.3 *Link between Presynaptic Release and Synaptic Stability*

Our model suggests that increasing the stability of synapses with large release probability improves memory. Qualitatively, this is in line with observations that presynaptic boutons, which contain stationary mitochondria [126, 127], are more stable than those which do not, both on short [128] and long timescales of at least weeks [129]. Quantitatively, we find evidence for such a link by re-analysing data[3] from Sjöström, Turrigiano & Nelson [130] for a spike-timing-dependent plasticity protocol in the rat primary visual cortex: Figure 4.9 of the supplementary material shows that synapses with higher initial release probability are more stable than those with low release probabilities for both LTP and LTD.

4.5.2.4 *Credit Assignment*

In our multilayer perceptron model, updates are computed using backpropagated gradients. Whether credit assignment in the brain relies on backpropagation – or more generally gradients – remains an active area of research, but several alternatives aiming to increase biological plausibility exist and are compatible with our model [131–133]. To check that the proposed mechanism can also operate without gradient information, we include an experiment with a standard perceptron and its gradient-free learning rule [107], see Figure 4.5b and 4.5c.

4.5.2.5 *Correspondence to Biological Networks*

We study general rate-based neural networks raising the question in which biological networks or contexts one might expect the proposed mechanisms to be at work. Our experiments suggest that improved energy efficiency can at least partly be attributed to the sparsification induced by presynaptic

3 Data was made publicly available in Costa *et al.* [117].

stochasticity [cf. 134]. Networks which are known to rely on sparse representations are thus natural candidates for the dynamics investigated here. This includes a wide range of sensory networks [135–138] as well as areas in the hippocampus [139, 140].

In the context of lifelong learning, our learning rule provides a potential mechanism that helps to slowly incorporate new knowledge into a network with preexisting memories. Generally, the introduced consolidation mechanism could benefit the slow part of a complementary learning system as proposed by [141, 142]. Sensory networks in particular might utilize such a mechanism as they require to learn new stimuli while retaining the ability to recognise previous ones [143–145]. Indeed, in line with the hypothesis that synapses with larger release probability are more stable, it has been observed that larger spines in the mouse barrel cortex are more stable. Moreover, novel experiences lead to the formation of new, stable spines, similar to our findings reported in Figure 4.8b.

4.5.3 *Related Synapse Models*

4.5.3.1 *Probabilistic Synapse Models*

The goal of incorporating and interpreting noise in models of neural computation is shared by many computational studies. Inspired by a Bayesian perspective, neural variability is often interpreted as representing uncertainty [146–149], or as a means to prevent overfitting [150]. The Bayesian paradigm has been applied directly to variability of individual synapses in neuroscience [151–153] and machine learning [110]. It prescribes decreasing the plasticity of synapses with low posterior variance. A similiar relationship can be shown to hold for our model as described in the Material and Methods. In contrast to common Bayesian interpretations [110, 148, 152] which model release statistics as Gaussians and optimize complex objectives [see also 90] our simple proposal represents the inherently discrete nature of synaptic transmission more faithfully.

4.5.3.2 *Complex Synapse Models*

In the context of lifelong learning, our model's consolidation mechanism is similar to EWC [11], which explicitly relies on the Fisher Information to consolidate synapses. Unlike EWC, our learning rule does not require a task switch signal and does not need a separate consolidation phase. Moreover, our model can be interpreted as using distinct states of plasticity to protect

memories. This general idea is formalised and analysed thoroughly by theoretical work on cascade models of plasticity [154–156]. The resulting model [156] has also been shown to be effective in lifelong learning settings [157].

4.5.4 *Synaptic Importance May Govern Energy-Information Trade-offs*

Energy constraints are widely believed to be a main driver of evolution [87]. From brain size [158, 159], to wiring cost [160], down to ion channel properties [161, 162], presynaptic transmitter release [94] and postsynaptic conductance [95, 96], various components of the nervous system have been shown to be optimal in terms of their total metabolic cost or their metabolic cost per bit of information transmitted.

Crucially, there is evidence that the central nervous system operates in varying regimes, making different trade-offs between synaptic energy demand and information transmission: Perge *et al.* [163], Carter & Bean [164], and Hu & Jonas [165] all find properties of the axon (thickness, sodium channel properties), which are suboptimal in terms of energy per bit of information. They suggest that these inefficiencies occur to ensure fast transmission of highly relevant information.

We propose that a similar energy/information trade-off could govern network dynamics preferentially allocating more energy to the most relevant synapses for a given task. Our model relies on a simple, theoretically justified learning rule to achieve this goal and leads to overall energy savings. Neither the trade-off nor the overall savings can be accounted for by previous frameworks for energy-efficient information transmission at synapses [89, 92].

This view of release probabilities and related metabolic cost provides a way to make the informal notion of "synaptic importance" concrete by measuring how much energy is spent on a synapse. Interestingly, our model suggests that this notion is helpful beyond purely energetic considerations and can in fact help to maintain memories during lifelong learning.

4.6 MATERIALS AND METHODS

4.6.1 *Summary of Learning Rule*

Our learning rule has two components, an update for the presynaptic release probability p_i and an update for the postsynaptic strength m_i. The

update of the synaptic strength m_i is defined implicitly through updating the expected synaptic strength $\bar{w}$

$$\bar{w}_i^{(t+1)} = \bar{w}_i^{(t)} - \eta g_i, \qquad \text{where } g_i = \frac{\partial L(\bar{w}^{(t)}, p^{(t)})}{\partial \bar{w}_i^{(t)}} \tag{4.6}$$

and the presynaptic update is given by

$$p_i^{(t+1)} = \begin{cases} p_i^{(t)} + p_{\text{up}}, & \text{if } |g_i| > g_{\text{lim}}, \\ p_i^{(t)} - p_{\text{down}}, & \text{if } |g_i| \leq g_{\text{lim}}. \end{cases} \tag{4.7}$$

This leads to the following explicit update rule for the synaptic strength $m_i = \frac{\bar{w}_i}{p_i}$

$$m_i^{(t+1)} = \frac{1}{p_i^{(t+1)}} \left(p_i^{(t)} m_i^{(t)} - \eta g_i \right) \tag{4.8}$$

$$= \frac{p_i^{(t)}}{p_i^{(t+1)}} m_i^{(t)} - \frac{\eta}{p_i^{(t+1)} p_i^{(t)}} \frac{\partial L(m^{(t)}, p^{(t)})}{\partial m_i^{(t)}} \tag{4.9}$$

where we used the chain rule to rewrite $g_i = \frac{\partial L}{\partial \bar{w}_i} = \frac{\partial L}{\partial m_i} \cdot \frac{\partial m_i}{\partial \bar{w}_i} = \frac{\partial L}{\partial m_i} \cdot \frac{1}{p_i}$.
For the lifelong learning experiment, we additionally stabilise high release probability synapses by multiplying the learning rate by $(1 - p_i)$ for each synapse and by freezing release probabilities (but not strengths) when they surpass a predefined threshold p_{freeze}.

4.6.2 *Theoretical Analysis of Presynaptic Learning Rule*

As indicated in the results section the release probability p_i is more likely to be large when the Fisher Information of the synaptic strength w_i is large as well. This provides a theoretical explanation to the intuitive correspondence between release probability and synaptic importance. Here, we formalise this link starting with a brief review of the Fisher Information.

4.6.2.1 *Link between Release Probabilities and Fisher Information*

We now explain how our learning rule for the release probability is related to the Fisher Information. For simplicity of exposition, we focus our analysis on a particular sampled subnetwork with deterministic synaptic strengths.

Recall that update rule (4.4) for release probabilities increases the release probability, if the gradient magnitude $|g_i|$ is above a certain threshold, $g_i > |g_{\text{lim}}|$, and decreases them otherwise. Let us denote by p_i^+ the probability that the i-th release probability is increased. Then

$$p_i^+ := \Pr[|g_i| > g_{\text{lim}}] = \Pr[g_i^2 > g_{\text{lim}}^2], \tag{4.10}$$

where the probability space corresponds to sampling training examples. Note that $\mathbb{E}[g_i^2] = EF_i$ by definition of the Empirical Fisher Information EF_i. So if we assume that $\Pr[g_i^2 > g_{\text{lim}}^2]$ depends monotonically on $\mathbb{E}[g_i^2]$, then we already see that p_i^+ depends monotonically on F_i. This in turn implies that synapses with a larger Fisher Information are more likely to have a large release probability, which is what we claimed. We now discuss the assumption made above.

ASSUMPTION: $\Pr[g_i^2 > g_{\text{LIM}}^2]$ DEPENDS MONOTONICALLY ON $\mathbb{E}[g_i^2]$. While this assumption is not true for arbitrary distributions of g, it holds for many commonly studied parametric families and seems likely to hold (approximately) for realistic, non-adversarially chosen distributions. For example, if each g_i follows a normal distribution $g_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ with varying σ_i and $\sigma_i \gg \mu_i$, then

$$F_i = \mathbb{E}[g_i^2] \approx \sigma_i^2$$

and

$$p_i^+ = \Pr[g_i^2 > g_{\text{lim}}^2] \approx \operatorname{erfc}\left(\frac{g_{\text{lim}}}{\sigma_i \sqrt{2}}\right)$$

so that p_i^+ is indeed monotonically increasing in F_i. Similar arguments can be made for example for a Laplace distribution, with scale larger than mean.

LINK BETWEEN LEARNING RATE MODULATION AND BAYESIAN UPDATING Recall that we multiply the learning rate of each synapse by $(1 - p_i)$, see equation (4.5). This learning rate modulation can be related to the update prescribed by Bayesian modelling. As shown before, synapses with large Fisher Information tend to have large release probability, which results in a decrease of the plasticity of synapses with large Fisher Information. We can treat the (diagonal) Fisher Information as an approximation of the posterior precision based on a Laplace approximation of the posterior likelihood [11] which exploits that the Fisher Information approaches the Hessian of the loss as the task gets learned [166]. Using this relationship,

our learning rate modulation tends to lower the learning rate of synapses with low posterior variance as prescribed by Bayesian modelling.

PRACTICAL APPROXIMATION The derivation above assumes that each gradient g is computed using a single input, so that $\mathbb{E}[g^2]$ equals the Fisher Information. While this may be the biologically more plausible setting, in standard ANN training the gradient is averaged across several inputs (mini-batches). Despite this modification, g^2 remains a good, and commonly used, approximation of the Fisher, see e.g. Supplementary 3.7.3.

4.6.3 *Perceptron for Lifelong Learning*

To demonstrate that our findings on presynaptic stochasticity and plasticity are applicable to other models and learning rules, we include experiments for the standard perceptron [107] in a lifelong learning setting.

4.6.3.1 *Model*

The perceptron is a classical model for a neuron with multiple inputs and threshold activation function. It is used to memorise the binary labels of a number of input patterns where input patterns are sampled uniformly from $\{-1, 1\}^N$ and their labels are sampled uniformly from $\{-1, 1\}$. Like in ANNs, the output neuron of a perceptron computes a weighted sum of its inputs followed by nonlinear activation $\sigma(\cdot)$:

$$a^{\text{post}} = \sigma\left(\sum_{i=1}^{n} w_i a_i^{\text{pre}}\right). \tag{4.11}$$

The only difference to the ANN model is that the nonlinearity is the sign function and that there is only one layer. We model each synapse w_i as a Bernoulli variable r_i with synaptic strength m_i and release probability p_i just as before, see equation (4.1). The expected strengths $\bar{w}_i$ are learned according to the standard perceptron learning rule [107]. The only modification we make is averaging weight updates across 5 inputs, rather than applying an update after each input. Without this modification, the update size g_i for each weight w_i would be constant according to the perceptron learning rule. Consequently, our update rule for p_i would not be applicable. However, after averaging across 5 patterns we can apply the same update rule for p_i as previously, see equation (4.4), and also use the same learning rate modification, see equation (4.5). We clarify that g_i now refers to the

update of expected strength $\bar{w}_i$. In the case of ANNs this is proportional to the gradient, while in the case of the non-differentiable perceptron it has no additional interpretation.

4.6.3.2 *Experiments*

For the lifelong learning experiments, we used 5 tasks, each consisting of 100 randomly sampled and labelled patterns of size $N = 1000$. We compared the perceptron with learned stochastic weights to a standard perceptron. For the standard perceptron, we also averaged updates across 5 patterns. Both models were sequentially trained on 5 tasks, using 25 passes through the data for each task.

We note that for more patterns, when the perceptron gets closer to its maximum capacity of $2N$, the average accuracies of the stochastic and standard perceptron become more similar, suggesting that the benefits of stochastic synapses occur when model capacity is not fully used.

As metaplasticity parameters we used $g_{\text{lim}} = 0.1, p_{\text{up}} = p_{\text{down}} = 0.2$ and $p_{\text{min}} = 0.25, p_{\text{freeze}} = 0.9$. These were coarsely tuned on an analogous experiment with only two tasks instead of five.

4.6.4 *Experimental Setup*

4.6.4.1 *Code Availability*

Code for all experiments is publicly available at
github.com/smonsays/presynaptic-stochasticity.

4.6.4.2 *Metaplasticity Parameters*

Our method has a number of metaplasticity parameters, namely p_{up}, p_{down}, g_{lim} and the learning rate η. For the lifelong learning experiments, there is an additional parameter p_{freeze}.
For the energy experiments we fix $p_{\text{up}} = p_{\text{down}} = 0.07$, $g_{\text{lim}} = 0.001$ and choose $\eta = 0.05$ based on coarse, manual tuning. For the lifelong learning experiments we choose $\eta_0 \in \{0.01, 0.001\}$ and optimise the remaining metaplasticity parameters through a random search on one task, namely Permuted MNIST, resulting in $p_{\text{up}} = 0.0516$, $p_{\text{down}} = 0.0520$ and $g_{\text{lim}} = 0.001$. We use the same fixed parametrisation for all other tasks, namely Permuted Fashion MNIST, Split MNIST and Split Fashion MNIST (see below for detailed task descriptions).

For the ablation experiment in Figure 4.5a, metaplasticity parameters were re-optimised for each ablation in a random search to ensure a fair, meaningful comparison.

4.6.4.3 *Model Robustness*

We confirmed that the model is robust with respect to the exact choice of parameters. For the energy experiments, de- or increasing $p_{\text{up}}, p_{\text{down}}$ by 25% does not qualitatively change results.
For the lifelong learning experiment, the chosen tuning method is a strong indicator of robustness: The metaplasticitiy parameters are tuned on one setup (Permuted MNIST) and then transferred to others (Split MNIST, Permuted & Split Fashion MNIST). The results presented in Table 4.1 show that the parameters found in one scenario are robust and carry over to several other settings. We emphasise that the differences between these scenarios are considerable. For example, for permuted MNIST consecutive input distributions are essentially uncorrelated by design, while for Split (Fashion) MNIST input distributions are strongly correlated. In addition, from MNIST to Fashion MNIST the number of "informative" pixels changes drastically.

4.6.4.4 *Lifelong Learning Tasks*

For the lifelong learning experiments we tested our method as well as baselines in several scenarios on top of the Split MNIST protocol described in the main text.

PERMUTED MNIST In the Permuted MNIST benchmark, each task consists of a random but fixed permutation of the input pixels of all MNIST images [10]. We generate 10 tasks using this procedure and present them sequentially without any indication of task boundaries during training. A main reason to consider the Permuted MNIST protocol is that it generates tasks of equal difficulty.

PERMUTED & SPLIT FASHION MNIST Both the Split and Permuted protocol can be applied to other datasets. We use them on the Fashion MNIST dataset [167] consisting of $60,000$ greyscale images of 10 different fashion items with 28×28 pixels.

CONTINUOUS PERMUTED MNIST We carry out an additional experiment on the continuous Permuted MNIST dataset [110]. This is a modified version of the Permuted MNIST dataset which introduces a smooth transition period between individual tasks where data from both distributions is mixed. It removes the abrupt change between tasks and allows us to investigate if our method depends on such an implicit task switch signal. We observe a mean accuracy over all tasks of 0.8539 ± 0.006 comparable to the non-continuous case suggesting that our method does not require abrupt changes from one task to another.

4.6.4.5 *Neural Network Training*

Our neural network architecture consists of two fully connected hidden layers of 200 neurons without biases with rectified linear unit activation functions $\sigma(x)$. The final layer uses a softmax and cross-entropy loss. Network weights were initialised according to the PyTorch default for fully connected layers, which is similar to Kaiming uniform initialisation [168, 169] but divides weights by an additional factor of $\sqrt{6}$. We use standard stochastic gradient descent to update the average weight $\bar{w}_i$ only altered by the learning rate modulation described for the lifelong learning experiments. We use a batch size of 100 and train each task for 10 epochs in the lifelong learning setting. In the energy-information experiments we train the model for 50 epochs.

4.7 SUPPLEMENTARY FIGURES AND TABLES

	Split MNIST	Split Fashion	Perm. MNIST	Perm. Fashion
Pre. Cons.	$82.90^{\pm 0.01}$	$91.98^{\pm 0.12}$	$86.14^{\pm 0.67}$	$75.92^{\pm 0.37}$
No Cons.	$77.68^{\pm 0.31}$	$88.76^{\pm 0.45}$	$79.60^{\pm 0.43}$	$72.13^{\pm 0.75}$
BGD	$80.44^{\pm 0.45}$	$89.54^{\pm 0.88}$	$89.73^{\pm 0.52}$	$78.45^{\pm 0.15}$
EWC	$70.41^{\pm 4.20}$	$76.89^{\pm 1.05}$	$89.58^{\pm 0.53}$	$77.44^{\pm 0.41}$
Joint	$98.55^{\pm 0.10}$	$97.67^{\pm 0.09}$	$97.33^{\pm 0.08}$	$87.33^{\pm 0.07}$

TABLE 4.1: **Lifelong Learning Comparison on Additional Datasets.** Average test accuracies (%, higher is better, average over all sequentially presented tasks) and standard errors for three repetitions of each experiment on four different lifelong learning tasks for the Presynaptic Consolidation mechanism, BGD [110] and EWC [11]. For the control "Joint Training" the network is trained on all tasks simultaneously serving as an upper bound of practically achievable performance.

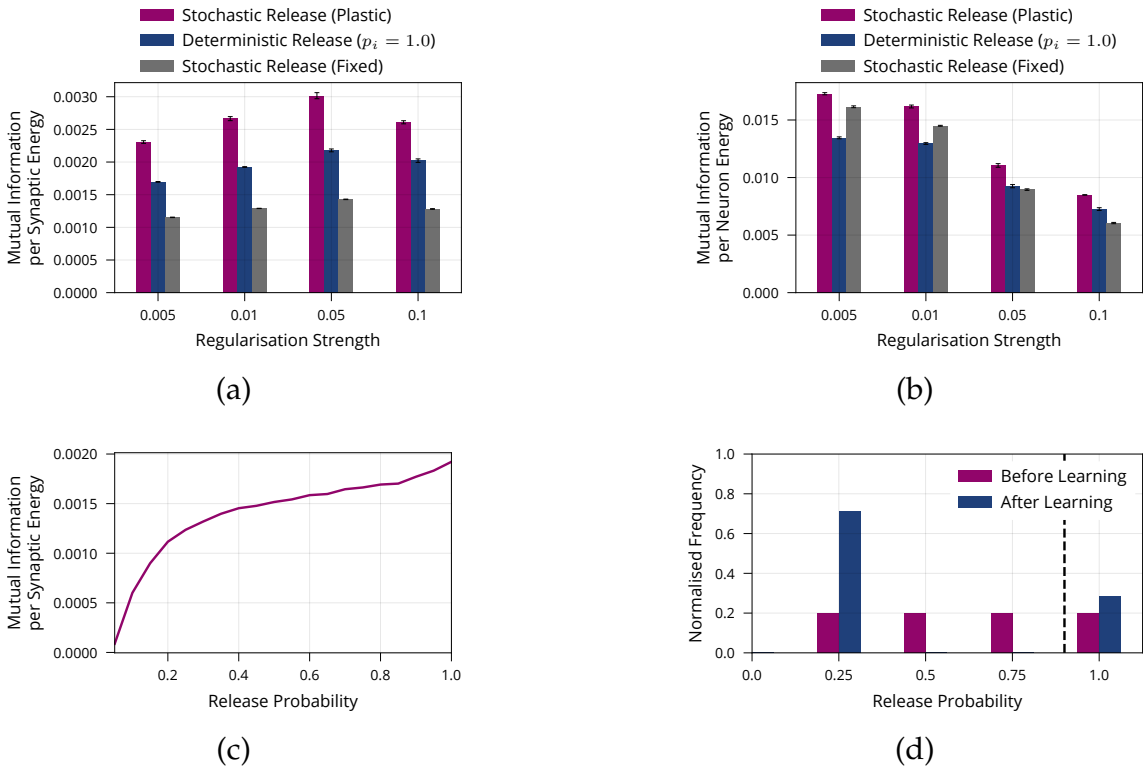
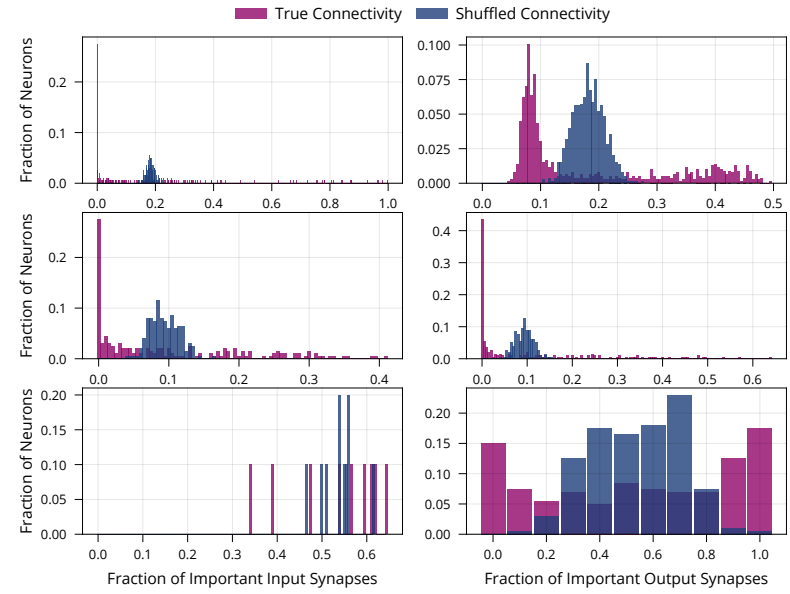
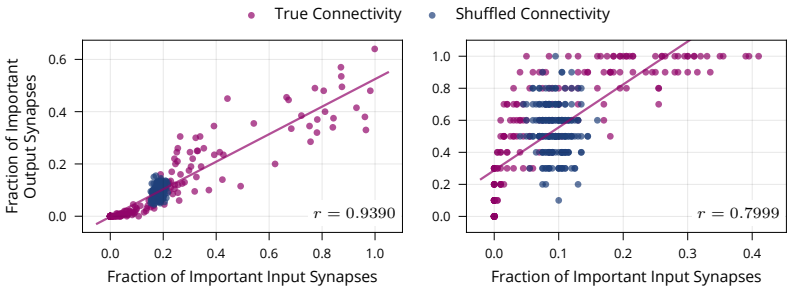


FIGURE 4.6: **Additional Results on Energy Efficiency of Model with Stochastic and Plastic Release. (a)** Mutual information per energy analogous to Figure 4.1b, but showing results for different regularisation strengths rather than the best result for each model. As described in the main part, energy is measured via its synaptic contribution. **(b)** Same experiment as in (a) but energy is measured as the metabolic cost incurred by the activity of neurons by calculating their average rate of activity. **(c)** Maximum mutual information per energy for a multilayer perceptron with fixed release probability and constant regularisation strength of 0.01. This is the same model as "Stochastic Release (Fixed)" in (a), but for a range of different values for the release probability. This is in line with the single synapse analysis in Harris, Jolivet & Attwell [89]. For each model, we searched over different learning rates and report the best result. **(d)** Analogous to Figure 4.2a, but release probabilities were initialised independently, uniformly at random in the interval $[0.25, 1]$ rather than with a fixed value of 0.25. Error bars in (a) and (b) denote the standard error for three repetitions of the experiment. (c) shows the best performing model for each release probability after a grid search over the learning rate. (d) shows aggregated data over three repetitions of the experiment.



(a)



(b)

FIGURE 4.7: **Additional Results on Neuron-Level Sparsity of Network after Learning. (a)** Number of important synapses per neuron for all layers after learning on MNIST. The i-th row shows data from the i-th weight matrix of the network and we compare true connectivity to random connectivity. Two-sample Kolmogorov-Smirnov tests comparing the distribution of important synapses in the shuffled and unaltered condition are significant for all layers ($p < 0.01$) except for the output neurons in the last layer (lower-left panel) ($p = 0.41$). This is to be expected as all 10 output neurons in the last layer should be equally active and thus receive similar numbers of active inputs. **(b)** Scatter plot showing the number of important input and output synapses per neuron for both hidden layers after learning on MNIST. First hidden layer (left) has a Pearson correlation coefficient of $r = 0.9390$. Second hidden layer (right) has a Pearson correlation coefficient of $r = 0.7999$. Data is from one run of the experiment.

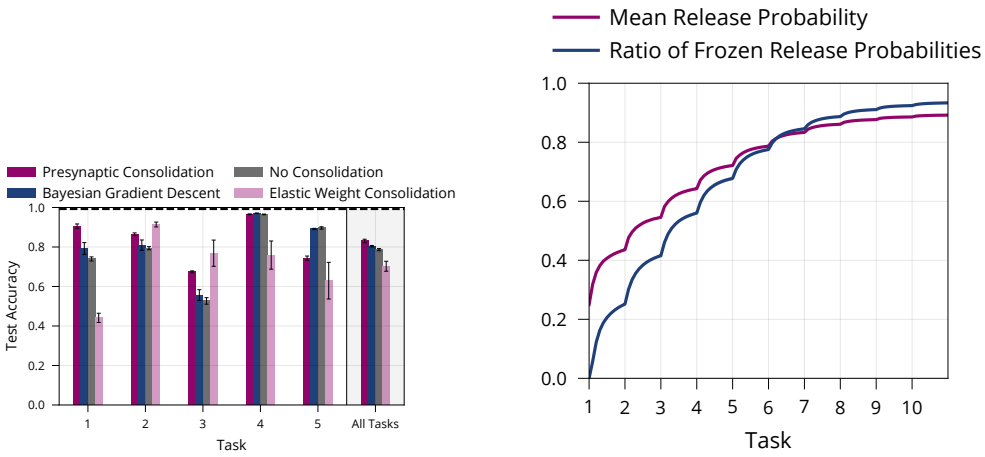


FIGURE 4.8: **Additional Results on Lifelong Learning in a Model with Presynaptically Driven Consolidation. (a)** Detailed lifelong-learning results of various methods on Split MNIST, same underlying experiment as in Figure 4.4c. We report the test accuracy on each task of the final model (after learning all tasks). Error bars denote the standard error for three repetitions of the experiment. **(b)** Mean release probability and percentage of frozen weights over the course of learning ten permuted MNIST tasks. Error bars in (a) and shaded regions in (b) show standard error over three repetitions of the experiment.

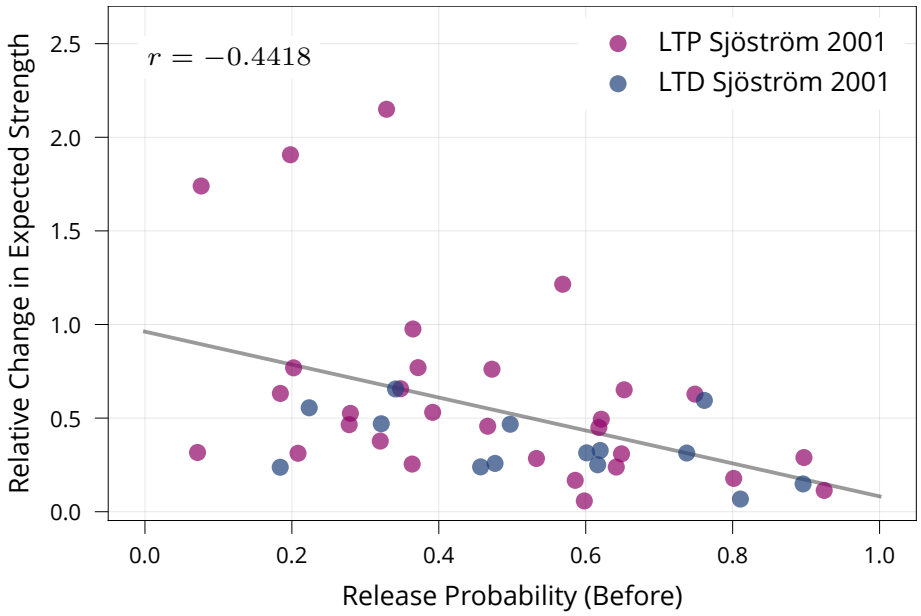


FIGURE 4.9: **Biological Evidence for Stability of Synapses with High Release Probability.**
To test whether synapses with high release probability are more stable than synapses with low release probability as prescribed by our model, we re-analysed data of [130] from a set of spike-timing-dependent plasticity protocols. The protocols induce both LTP and LTD depending on their precise timing. The figure shows that synapses with higher release probabilities undergo smaller relative changes in expected strength (Pearson Corr. $r = -0.4416$, $p < 0.01$) . This suggests that synapses with high release probability are more stable than synapses with low release probability, matching our learning rule.

5

GRADIENT DESCENT ON NEURONS AND ITS LINK TO APPROXIMATE SECOND-ORDER OPTIMIZATION

This chapter is taken and partially adapted from Benzing [44], which will have been appeared at ICML 2022.

5.1 INTRODUCTION

As we've described in Chapters 1 and 2 a natural tool to get a better understanding of the local properties of loss landscapes is the second-order Taylor approximation and one way to deal with the prohibitive memory cost of computing the Fisher (or Hessian) is to approximate it.

A family of approximations that has been particularly successful are Kronecker-factored, block diagonal approximations of the curvature, as introduced in Chapter 2. They were originally proposed in the context of optimization as KFAC [41] and concurrently as Natural Neural Networks [42]. Below, for simplicity, we will mostly refer to and compare to the former variant, KFAC, and our findings apply similarly Natural Neural Networks. In fact, Section 5.4 can almost be seen as re-deriving the update of Natural Neural Networks from a different perspective.

Within the optimization community Kronecker-factored optimizers have lead to many further developments [170–177], and they have also proven influential in various other contexts like Bayesian inference, meta learning and continual learning [178–183].

In this chapter, we describe a surprising discovery: Despite its motivation, the KFAC optimizer does not rely on second-order information; in particular it significantly outperforms exact second-order optimizers. We establish these claims through a series of careful ablations and control experiments and build on prior work, which shows that exact second-order updates can be computed efficiently and exactly, if the dataset is small or when the curvature matrix is subsampled [40, 53].

Our finding that KFAC does not rely on second-order information immediately raises the question why it is nevertheless so effective. To answer this question, we present evidence that KFAC approximates a different, first-order optimizer, which performs gradient descent in neuron- rather

than weight space. We also show that this optimizer itself often improves upon KFAC, both in terms of computational cost as well as progress per parameter update.

5.2 NOTATION AND TERMINOLOGY

In this chapter, we typically focus on one layer of a neural network. For simplicity of notation, we consider fully-connected layers, but results can easily be extended to architectures with parameter sharing, like CNNs or RNNs.

We denote the layer's weight matrix by $\mathbf{W} \in \mathbb{R}^{n \times m}$ and its input-activations (after the previous' layer nonlinearity) by $\mathbf{A} \in \mathbb{R}^{m \times D}$, where D is the number of datapoints. The layer's output activations (before the nonlinearity) are equal to $\mathbf{B} = \mathbf{W}\mathbf{A} \in \mathbb{R}^{n \times D}$ and we denote the partial derivates of the loss L with respect to these outputs (usually computed by backpropagation) by $\mathbf{E} = \frac{\partial L}{\partial \mathbf{B}}$. If the label is sampled from the model's output distribution, as is the case for the Fisher (2.5), we will use $\mathbf{E}_F$ rather than $\mathbf{E}$.

We use the term "**datapoint**" for a pair of input and label (X, y). In the context of the Fisher information, the label will always be sampled from the model's output distribution. Note that with this definition, the total number of datapoints is the product of the number of inputs and the number of labels.

Following Martens & Grosse [41], the Fisher will usually be approximated by sampling one label for each input. For some controls, we will distinguish whether one label is sampled or whether the full Fisher is computed, and we will refer to the former as **MC Fisher** and the latter as **Full Fisher**.

As is common in the ML context, we will use the term "**second-order method**" for algorithms that use (approximate) second derivatives. The term "**first-order method**" will refer to algorithms which only use first derivatives or quantities that are independent of the loss, i.e. "zero-th" order terms.

5.3 EXACT NATURAL GRADIENTS AND KFAC

In this section, we will show that KFAC is not strongly related to second-order information. We start with a brief review of KFAC and then proceed with experiments.

5.3.1 *Review of KFAC*

KFAC makes two approximations to the Fisher as also detailed in Chapter 2. Firstly, it only considers diagonal blocks of the Fisher, where each block corresponds to one layer of the network. Secondly, each block is approximated as a Kronecker product $(\mathbf{A}\mathbf{A}^T) \otimes (\mathbf{E}_F\mathbf{E}_F^T)$. This approximation of the Fisher leads to the following approximate natural gradient update.

$$(\Delta\mathbf{W})^T = \left(\mathbf{A}\mathbf{A}^T + \lambda_A\mathbf{I}\right)^{-1}\left(\mathbf{A}\mathbf{E}^T\right)\left(\mathbf{E}_F\mathbf{E}_F^T + \lambda_E\mathbf{I}\right)^{-1} \tag{5.1}$$

where λ_A, λ_E are damping terms satisfying $\lambda_A \cdot \lambda_E = \lambda$ for a hyperparameter λ and $\frac{\lambda_A}{\lambda_E} = \frac{n \cdot \mathrm{Tr}(\mathbf{A}\mathbf{A}^T)}{m \cdot \mathrm{Tr}(\mathbf{E}_F\mathbf{E}_F^T)}$.

Heurisitc Damping
We emphasise that the damping performed here is heuristic: Every Kronecker factor is damped individually. This deviates from the theoretically "correct" form of damping, which consists of adding a multiple of the identity to the approximate curvature. To make this concrete, the two strategies use the following damped curvature matrices

$$\text{standard:} \quad \left(\mathbf{A}\mathbf{A}^T \otimes \mathbf{E}_F\mathbf{E}_F^T\right) + \lambda\mathbf{I} \tag{5.2}$$

$$\text{heuristic:} \quad \left(\mathbf{A}\mathbf{A}^T + \lambda_A\mathbf{I}\right) \otimes \left(\mathbf{E}_F\mathbf{E}_F^T + \lambda_E\mathbf{I}\right) \tag{5.3}$$

Heuristic damping adds undesired cross-terms $\lambda_E\mathbf{A}\mathbf{A}^T \otimes \mathbf{I}$ and $\lambda_A\mathbf{I} \otimes \mathbf{E}_F\mathbf{E}_F^T$ to the curvature, and we point out that these cross terms are typically much larger than the desired damping $\lambda\mathbf{I}$. While the difference in damping may nevertheless seem innocuous, Martens & Grosse [41], Ba, Grosse & Martens [171], and George *et al.* [173] all explicitly state that heuristic damping performs better than standard damping. From a theoretical perspective, this is a rather mysterious observation.

In practice, the Kronecker factors $\mathbf{A}\mathbf{A}^T$ and $\mathbf{E}_F\mathbf{E}_F^T$ are updated as exponential moving averages, so that they incorporate data from several recent mini-batches.

Subsampled Natural Gradients vs KFAC
There are two high level differences between KFAC and subsampled natural gradients. (1) KFAC can use more data to estimate the Fisher, due to its exponential moving averages. (2) For a given mini-batch, natural gradients are exact, while KFAC makes additional approximations.

A priori, it seems that (1) is a disadvantage for subsampled natural gradients, while (2) is an advantage. However, we will see that this is not the case.

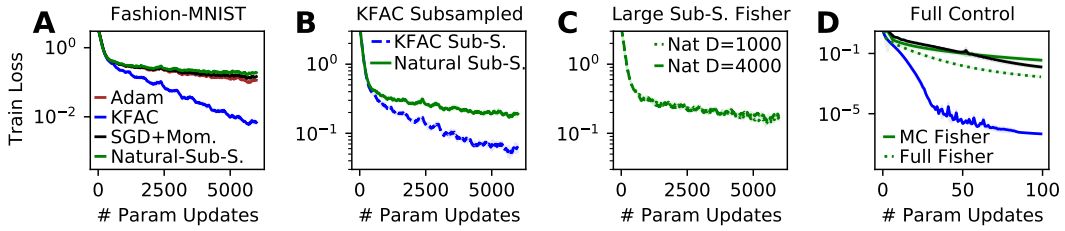


FIGURE 5.1: **Paradoxically, KFAC – an approximate second-order method – outperforms exact second-order udpates in standard as well as important control settings. (A)** Comparison between Subsampled Natural Gradients and KFAC. KFAC performs significantly better. Theoretically, its only advantage over the subsampled method is using more data to estimate the curvature. All methods use a batchsize of 100 and are trained for 10 epochs, with hyperparameters tuned individually for each method (here and in all other experiments). **(B)** Comparison between Subsampled Natural Gradients and Subsampled KFAC. Both algorithms use exactly the same amount of data to estimate the curvature. From a theoretical viewpoint, KFAC should be a strictly worse approximation of second-order updates than the exact subsampled method; nevertheless, it performs significantly better. **(C)** Additional control in which the subsampled Fisher is approximated on larger mini-batches. **(D)** Full control setting, in which the training set is restricted to 1000 images and gradients and curvature are computed on the entire batch (in addition, for a clean comparison KFAC does not use an exponential average to estimate the curvature). The dashed green line corresponds to exact natural gradients without any approximations. Consistent with prior literature, full second-order updates do outperform standard first-order updates (dashed green vs. black line). More importantly, and very surprisingly, KFAC significantly outperforms exact second-order updates. This is very strong evidence that KFAC is not closely related to Natural Gradients.
(A-D) We repeat several key experiments with other datasets and architectures and results are consistent with the ones seen here, see main text and Supplementary Material. **(A-D)** Solid lines show mean across three seeds; shaded regions (here and in remaining main chapter figures) show mean$\pm$std, but for most experiments are visually hard to distinguish from the mean.

5.3.2 *Experiments*

The first set of experiments is carried out on a fully connected network on Fashion MNIST [184] and followed by results on a Wide ResNet [28] on CIFAR10 [185]. We run several additional experiments, which are presented fully in the Supplementary Material, and will be refered to in the main text. These include repeating the first set of experiments on MNIST; results on CIFAR100; a VGG network [186] trained on SVHN [187] and more traditional autoencoder experiments [188].

We emphasise that, while our results are surprising, they are certainly not caused by insufficient hyperparameter tuning or incorrect computations of second-order updates. In particular, we perform independent grid searches for each method and ablation and make sure that the grids are sufficiently wide and fine. Details are given in Appendix A and D of Benzing [44] and part of the software validation in Appendix B of Benzing [44]. Code to validate and run the software is provided on github. Moreover, as will be pointed out throughout the text, our results are consistent with many experiments from prior work.

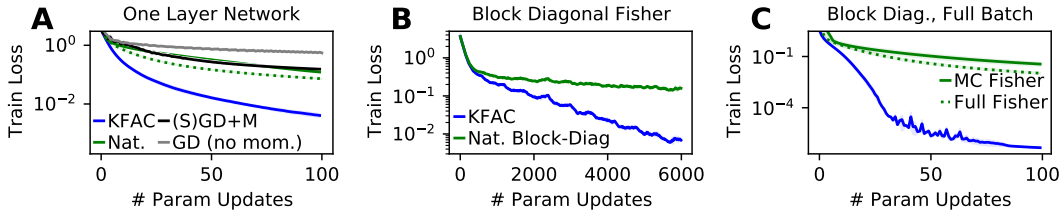


FIGURE 5.2: **Advantage of KFAC over exact, subsampled Natural Gradients is not due to block-diagonal structure. (A)** A one layer network (i.e. we perform logistic regression) is trained on 1000 images and full batch gradients are used. In particular, KFAC and the subsampled method use the same amount of data to estimate the curvature. In a one layer network the block-diagonal Fisher coincides with the full Fisher, but KFAC still clearly outperforms natural gradients. **(B)** Comparison between KFAC and layerwise (i.e. block-diagonal) subsampled Natural Gradients on full dataset with a three layer network. **(C)** Same as (B), but training set is restricted to a subset of 1000 images and full-batch gradient descent is performed. **(A-C)**Experiments on Fashion MNIST, results on MNIST are analogous, see Supplementary Material.

To obtain easily interpretable results without unnecessary confounders, we choose a constant step size for all methods, and a constant damping term.

This matches the setup of prior work [42, 173, 177, 189]. We re-emphasise that these hyperparameters are optimized carefully and indepently for each method and experiment individually.

Following the default choice in the KFAC literature [41], we usually use a Monte Carlo estimate of the Fisher, based on sampling one label per input. We will also carry out controls with the Full Fisher.

Performance

We first investigate the performance of KFAC and subsampled natural gradients, see Figure 5.1A. Surprisingly, natural gradients significantly underperform KFAC, which reaches an approximately 10-20x lower loss on both Fashion MNIST and MNIST. This is a concerning finding, requiring further investigation: After all, the exact natural gradient method should in theory perform at least as good as any approximation of it. Theoretically, the only potential advantage of KFAC over subsampled natural gradients is that it uses more information to estimate the curvature.

Controlling for Amount of Data used for the Curvature

The above directly leads to the hypothesis that KFAC's advantage over subsampled natural gradients is due to using more data for its approximation of the Fisher. To test this hypothesis, we perform three experiments. (1) We explicitly restrict KFAC to use the same amount of data to estimate the curvature as the subsampled method. (2) We allow the subsampled method to use larger mini-batches to estimate the Fisher. (3) We restrict the training set to 1000 (randomly chosen) images and perform full batch gradient descent, again with both KFAC and subsampled natural gradients using the same amount of data to estimate the Fisher. Here, we also include the Full Fisher information as computed on the 1000 training samples, rather than simply sampling one label per datapoint (MC Fisher). In particular, we evaluate exact natural gradients (without any approximations: The gradient is exact, the Fisher is exact and the inversion is exact). The results are shown in Figure 5.1 and all lead to the same conclusion: The fact that KFAC uses more data than subsampled natural gradients does not explain its better performance. In particular, subsampled KFAC outperforms exact natural gradients, also when the latter can be computed without any approximations.

This first finding is very surprising. Nevertheless, we point out that it is consistent with experimental results from prior work as well as commonly held beliefs. Firstly, it is widely believed that subsampling natural gradients leads to poor performance. This belief is partially evidenced by claims from Martens *et al.* [39] and often mentioned in informal discussions and

reviews. It matches our findings and in particular Figure 5.1D, which shows that benefits of Natural Gradients over SGD only become notable when computing the Fisher fully.[1] Secondly, we have shown that KFAC performs well even when it is subsampled in the same way as we subsampled natural gradients. While this does seem to contradict the belief that natural gradient methods should not be subsampled, it is confirmed by experiments from Botev, Ritter & Barber [172] and Bernacchia, Lengyel & Hennequin [176]: See Figure 2 "per iteration curvature" in Botev, Ritter & Barber [172] and note that in Bernacchia, Lengyel & Hennequin [176] the curvature is evaluated on individual minibatches.

Additional Experiments
We repeat the key experiments from Figure 5.1A,B in several additional settings: On a MLP on MNIST, on a ResNet with and without batch norm on CIFAR10 and for traditional autoencoder experiments. The findings are in line with the ones above, and solidify concerns whether KFAC is related to second-order information.

Controlling for Block-Diagonal Structure
This begs further investigation into why KFAC outperforms natural gradients. KFAC approximates the Fisher as block-diagonal. To test whether this explains KFAC's advantage, we conduct two experiments. First, we train a one layer network on a subset of 1000 images with full-batch gradient descent (i.e. we perform logistic regression). In this case, the block-diagonal Fisher coincides with the Fisher. So, if the block-diagonal approximation were responsible for KFAC's performance, then for the logistic regression case, natural gradients should perform as well as KFAC or better. However, this is not the case as shown in Figure 5.2A. As an additional experiment, we consider a three layer network and approximate the Fisher by its block-diagonal (but without approximating blocks as Kronecker products). The resulting computations and inversions can be carried out efficiently akin to the subsampled natural gradient method. We run the block-diagonal natural gradient algorithm in two settings: In a minibatch setting, identical to the one shown in Figure 5.1 and in a full-batch setting, by restricting to a subset of 1000 training images. The results in Figure 5.2B,C confirm our previous findings: (1) KFAC significantly outperforms even exact block-diagonal natural gradients (with full Fisher and full gradients). (2) It is not the block-diagonal structure that explains KFAC's performance.

1 It also evidences the correctness of our implementation of natural gradients.

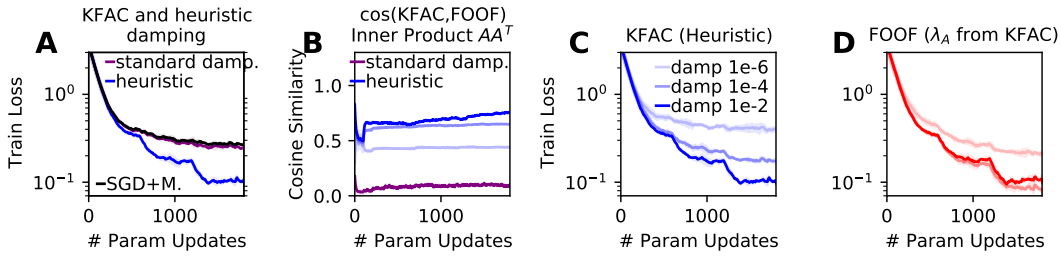


FIGURE 5.3: **Heuristic Damping increases KFAC's performance as well as its similarity to first-order method FOOF. (A)** Heuristic damping is strictly needed for performance of KFAC; with standard damping, KFAC performs similar to SGD. **(B)** Heuristic damping significantly increases similarity of KFAC to FOOF. For the inner product space, we use the "curvature" matrix of FOOF. **(C+D)** Performance of KFAC and FOOF across different damping strengths using heuristic damping for KFAC. For a clean and fair comparison, this version of FOOF uses λ_A from KFAC, see Appendix D.9 of Benzing [44] for full details. Notably, FOOF already works well for lower damping terms than KFAC, suggesting that KFAC requires larger damping mainly to guarantee similarity to FOOF and limit the effect of its second Kronecker factor. **(A-D)** and our theoretical analysis suggest that KFAC owes its performance to similarity to the first-order method FOOF. Experiments are on Fashion-MNIST, results on MNIST are analogous, see Supplementary Material. We also re-run experiment (A) in several other settings and confirm that heuristic damping is crucial for performance, see Appendix. This is in line with reports from [41, 171, 173].

Heuristic Damping

KFAC also deviates from exact second-order updates through its heuristic damping. To test whether this difference explains KFAC's performance, we implemented a version of KFAC with standard damping.[2] Figure 5.3A shows that KFAC owes essentially all of its performance to the damping heuristic. This finding is confirmed by experiments on CIFAR10 with a ResNet and on autoencoder experiments. We re-emphasise that KFAC outperforms exact natural gradients, and therefore the damping heuristic cannot be seen as giving a better approximation of second-order updates. Rather, heuristic damping causes performance benefits through some other effect.

2 This can be done with ideas from [173].

Summary

We have seen that KFAC, despite its motivation as an approximate natural gradient method, behaves very differently from true natural gradients. In particular, and surprisingly, KFAC drastically outperforms natural gradients. Through a set of careful controls, we established that KFAC's advantage relies on a seemingly innocuous damping heuristic, which is unrelated to second-order information. We now turn to why this is the case.

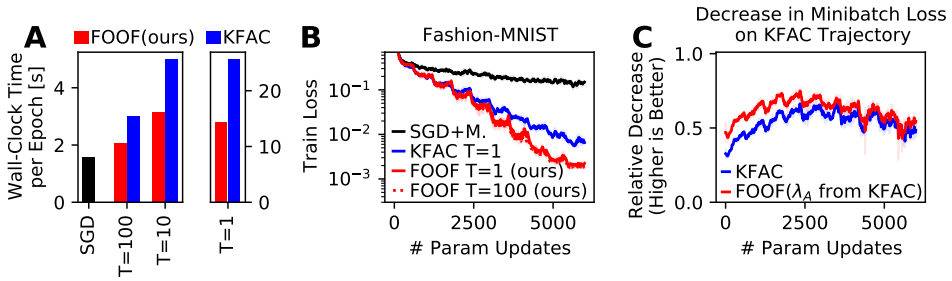


FIGURE 5.4: **FOOF outperforms KFAC in terms of both per-update progress and computation cost. (A)** Wall-clock time comparison between FOOF, KFAC and SGD. T denotes how frequently matrix inversions (see eq (5.5)) are performed. Implemented with PyTorch and run on a GPU. Increasing T above 100 does not notably improve runtime. FOOF is approximately 1.5x faster than KFAC. **(B)** Training loss on Fashion MNIST. FOOF is more data efficient and stable than KFAC. **(C)** Comparison of KFAC and a version of KFAC which drops the second Kronecker factor (equivalently, this corresponds to FOOF with damping term λ_A from KFAC). We follow the trajectory of KFAC, and at each point we compute the relative improvement of the loss on the current mini-batch that is achieved by the update of KFAC and FOOF, respectively. We use the learning rate and damping that is optimal for KFAC, and scale the FOOF update to have the same norm as the KFAC update at each layer. FOOF performs better, further suggesting that similarity to FOOF is responsible for KFAC's performance. **(B+C)** See also Supplementary Material Fig 5.17 for an instance, where FOOF makes more progress per update, but the overall KFAC trajectory performs better.

5.4 FIRST-ORDER DESCENT ON NEURONS

We will first describe the optimizer "Fast First-Order Optimizer" or "FOOF"[3] and then explain KFAC's link to it.

FOOF's update rule is similar to some prior work [42, 43, 190] and is also related to the idea of optimizing modules of a nested function independently [191–194] . The view on optimization which underlies FOOF is principled and new, and, among other differences, our insights and experiments linking KFAC to FOOF are new. For a more detailed discussion see Supplementary Material 5.7.2.5.

Recall our notation for one layer of a neural network from Section 5.2, namely $\mathbf{A}, \mathbf{W}$ for input activation and weight matrix as well as $\mathbf{B} = \mathbf{W}\mathbf{A}$ and $\mathbf{E} = \frac{\partial L}{\partial B}$.

Typically, for first-order optimizers, we compute the weights' gradients for each datapoint and average the results. Changing perspective, we can try to find an update of the weight matrix that explicitly changes the layer's outputs $\mathbf{B}$ into their gradient direction $\mathbf{E} = \frac{\partial L}{\partial \mathbf{B}}$. In other words, we want to find a weight update $\Delta\mathbf{W}$ to the parameters $\mathbf{W}$, so that the layer's output changes in the gradient direction, i.e. $(\mathbf{W} + \Delta\mathbf{W})\mathbf{A} = \mathbf{B} + \eta \frac{\partial L}{\partial \mathbf{B}}$ or equivalently $(\Delta\mathbf{W})\mathbf{A} = \eta\mathbf{E}$ for a learning rate η. Formally, we optimize

$$\min_{\Delta\mathbf{W} \in \mathbb{R}^{n \times m}} \|(\Delta\mathbf{W})\mathbf{A} - \eta\mathbf{E}\|^2 + \frac{\lambda}{2}\|\Delta\mathbf{W}\|^2 \tag{5.4}$$

where the second summand $\frac{\lambda}{2}\|\Delta\mathbf{W}\|^2$ is a proximity constraint limiting the update size. (5.4) is a linear regression problem (for each row of $\Delta\mathbf{W}$) solved by

$$(\Delta\mathbf{W})^T = \eta \left(\lambda\mathbf{I} + \mathbf{A}\mathbf{A}^T\right)^{-1} \mathbf{A}\mathbf{E}^T \tag{5.5}$$

Pseudocode for the resulting optimizer FOOF is presented in Supplementary Material 5.7.3.6. Figures 5.4,5.5 show the empirical results of FOOF, which outperforms not only SGD and Adam, but often also KFAC. An intuition for why "gradient descent on neurons" performs considerably better than "gradient descent on weights" is that it trades off conflicting gradients from different data points more effectively than the simple averaging scheme of SGD. See Appendix F of Benzing [44] for an illustrative toy example for this intuition.

3 F_2O_2 is a chemical also referred to as "FOOF".

The FOOF udpate can be seen as preconditioning by $\left((\lambda \mathbf{I} + \mathbf{A}\mathbf{A}^T) \otimes \mathbf{I}\right)^{-1}$ and we emphasise that this matrix contains no dependence on the loss, or first/second derivatives of it, so that it cannot be seen as a "second-order" optimizer according to common ML terminology.

5.4.1 *Stochastic Version of FOOF and Amortisation*

The above formulation is implicitly based on full-batch gradients. To apply it in a stochastic setting, we need to take some care to limit the bias of our updates. In particular, for the updates to be completely unbiased one would need to compute $\mathbf{A}\mathbf{A}^T$ for the entire dataset and invert the corresponding matrix at each iteration. This is of course too costly and instead we keep an exponentially moving average of mini-batch estimates of $\mathbf{A}\mathbf{A}^T$, which are computed during the standard forward pass. To amortise the cost of inverting this matrix, we only perform the inversion every T iterations. This leads to slightly stale values of the inverse, but in practice the algorithm is remarkably robust and allows choosing large values of T as also shown in Figure 5.4.

FOOF can be straightforwardly combined with momentum and (decoupled) weight decay.

5.4.2 *KFAC as First-Order Descent on Neurons*

Recall that the KFAC update is given by

$$(\Delta \mathbf{W})^T = \left(\mathbf{A}\mathbf{A}^T + \lambda_A \mathbf{I}\right)^{-1} \left(\mathbf{A}\mathbf{E}^T\right) \left(\mathbf{E}_F \mathbf{E}_F^T + \lambda_E \mathbf{I}\right)^{-1}.$$

Similarity of KFAC to FOOF and Damping

The update of KFAC differs from the FOOF update (eq (5.5)) only through the second factor $\left(\mathbf{E}_F \mathbf{E}_F^T + \lambda_E \mathbf{I}\right)^{-1}$. We emphasise that this similarity is induced mainly through the heuristic damping strategy. In particular, with standard damping, or without damping, the second Kronecker factor of KFAC could lead to updates that are essentially uncorrelated with FOOF. However, as we use heuristic damping and increase the damping strength λ_E, the second factor will be closer to (a multiple of) the identity and KFAC's update will become more and more aligned with FOOF.

Based on this derivation, we now test empirically whether heuristic damping indeed makes KFAC similar to FOOF and how it affects performance.

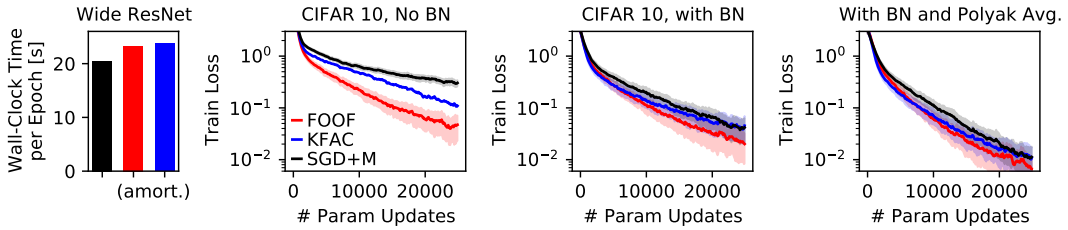


FIGURE 5.5: **FOOF outperforms KFAC in a Wide ResNet18 on CIFAR 10. (A)** Wall-clock time comparison between SGD and amortised versions of FOOF, KFAC. In convolutional architectures, FOOF and KFAC can be effectively amortised wihtout sacrificing performance. **(B, C, D)** Training loss in different settings. **(A-D)** Results on CIFAR100 and SVHN are analogous.

Figure 5.3B confirms our theoretical argument that heuristic damping drastically increases similarity of KFAC to FOOF and stronger heuristic damping leads to even stronger similarity. This similarity is directly linked to performance of KFAC as shown in Figure 5.3C. These findings, in particular the necessity to use heuristic damping, already strongly suggest that similarity to FOOF is required for KFAC to perform well. Moreover, as shown in Figure 5.3D, FOOF requires lower damping than KFAC to perform well. This further suggests that damping in KFAC is strictly required to limit the effect of $\mathbf{E}_F\mathbf{E}_F^T$ on the update, thus increasing similarity to FOOF. All in all, these results directly support the claim that KFAC, rather than being a natural gradient method, owes its performance to approximating FOOF.

Performance

If the above view of KFAC is correct, and it owes its performance to similarity to FOOF, then one would expect FOOF to perform better than or similarly to KFAC. We carry out two different tests of this hypothesis. First, we train a network using KFAC and at each iteration, we record the progress KFAC makes on the given mini-batch, measured as the relative decrease in loss. We compare this to the progress that KFAC would have made without its second Kronecker factor. We use learning rate and damping that are optimal for KFAC and, when dropping the second factor, we rescale the update to have the same norm as the original KFAC update for each layer. The results are shown in Figure 5.4C and show that without the second factor, KFAC makes equal or more progress, which supports our hypothesis. This observation is consistent across all different experimental setups we investigated, see Supplementary Material.

As a second test, we check whether FOOF outperforms KFAC, when both algorithms follow their own trajectory. This is indeed the case as shown in Figure 5.4B. The only case where the advantage described in Figure 5.4C does not translate to an overall better performance is the autoencoder setting, as analysed in the Supplementary Material, e.g. Figure 5.17, Sec 5.7.3.2. Results on a Wide ResNet18 demonstrate that our findings carry over to more complex settings and that FOOF often outperforms KFAC, see Figure 5.5.

Computational Cost
We also note that, on top of making more progress per parameter update, FOOF requires strictly less computation than KFAC: It does not require an additional backward pass to estimate the Fisher; it only requires keeping track of, inverting as well as multiplying the gradients by one matrix rather than two (only $\mathbf{A}\mathbf{A}^T$ and not $\mathbf{E}_F\mathbf{E}_F^T$). These savings lead to a 1.5x speed-up in wall-clock time per-update for the amortised versions of KFAC and FOOF as shown in Figure 5.4A.

Cost in Convolutional Architectures
The only overhead of KFAC and FOOF which cannot be amortised is performing the matrix multiplications in eqs (5.5),(5.1). These are standard matrix-matrix multiplications and are considerably cheaper than convolutions, so that we found that KFAC and FOOF can be amortised to have almost the same wall-clock time per update as SGD for this experiment ($\sim$10% increase for FOOF, $\sim$15% increase for KFAC) without sacrificing performance, see Appendix D of Benzing [44] for full details. We note that these results are significantly better than wall-clock times from Desjardins *et al.* [42] and Ba, Grosse & Martens [171], which require approximately twice as much time per update as SGD.[4]

Summary
We had already seen that KFAC does not rely on second-order information. In addition, these results suggest that KFAC owes its strong performance to its similarity to FOOF, a principled, well-performing first-order optimizer.

5.5 LIMITATIONS

In our experiments, we report training losses and tune hyperparameters with respect to them. While this is the correct way to test our hypotheses

4 Information reconstructed from Figure 3 in Ba, Grosse & Martens [171] and Figure 4 in Desjardins *et al.* [42].

and common for developing and testing optimizers [7], it will be important to test how well the optimizers investigated here generalise. A meaningful investigation of generalisation requires a different experimental setup (as demonstrated in Zhang *et al.* [189]) and is left to future work. With this in mind, we note that in our setting the advantage of FOOF and KFAC in training loss typically translates to an advantage in validation accuracy (and that FOOF and KFAC behave similarly).

We have restricted our investigation to the context of optimization and more specifically KFAC. While we strongly believe that our findings carry over to other Kronecker-factored optimizers [42, 172–175, 177], we have not explicitly tested this.

While, in all our experiments, the newly proposed view that KFAC is closely related to FOOF captures considerably more characteristics of KFAC than the standard view of KFAC as a natural gradient method, we highlight again that there are some limitations to this explanation and, in particular, one setting where KFAC performs slightly better than FOOF, see Section 5.7.3.2.

5.6 DISCUSSION

The purpose of this discussion is twofold. On the one hand, we will show that, while being surprising and contradicting common, strongly held beliefs, much of our results are consistent with data from prior work. On the other hand, we will summarise in how far our fundamentally new explanation for KFAC's effectiveness improves upon prior knowledge and resolves several puzzling observations.

Natural Gradients vs KFAC

Our first key result is that KFAC outperforms exact natural gradients, despite being motivated as an approximate natural gradient method. We perform several controls and a particularly important set of experiments is comparing exact, subsampled natural gradients to subsampled KFAC in a range of settings. In these experiments, we find: (1) Subsampled natural gradients often do not perform much better than SGD with momentum. (2) Subsampled KFAC works very well. Finding (1) is consistent with rather common beliefs that subsampling the curvature is harmful. These beliefs are often uttered in informal discussions and are partially evidenced by claims from Martens *et al.* [39]. Moreover, in some controls (e.g. Fig 5.1D), we show that full (non-subsampled) natural gradients do outperform SGD with momentum, consistent for example with Martens *et al.* [39]. Thus,

finding (1) is in line with prior knowledge and results. Moreoever, finding (2) matches experiments from Botev, Ritter & Barber [172] and Bernacchia, Lengyel & Hennequin [176] as described in Section 5.3 and thus also is in line with prior work.

Also independently of our results, it is worth noting that the performance of subsampled KFAC reported in Botev, Ritter & Barber [172] and Bernacchia, Lengyel & Hennequin [176] is hard to reconcile with the simultaneous convictions that (1) KFAC is a natural gradient method and (2) subsampling the Fisher has detrimental effects.

Our newly suggested explanation of KFACs performance resolves this contradiction. Even if one were to disagree with our explanation for KFAC's effectiveness, the above is an important insight, strengthened by our careful control experiments, and deserves further attention.

It is also worth noting that another natural way to check if our finding that KFAC outperforms Natural Gradients agrees with prior work would be to look for a direct comparison of KFAC with Hessian-Free optimization (HF). Perhaps surprisingly, to the best of our knowledge, there is no meaningful comparison between these two algorithms in the literature.[5] It will be interesting to see a thoroughly controlled, well tuned comparison between HF and KFAC.

Damping

A second cornerstone of our study is the effect of damping on KFAC. We found that employing a heuristic, rather than standard damping strategy is essential for performance. The result that heuristic damping improves KFAC's performance has been noted several times (qualitatively, rather than quantitatively), see the original KFAC paper [41], its large-scale follow up [171], and even E-KFAC [173].

While choice of damping strategy may seem like a negligible detail at first, it is important to bear in mind that without heuristic damping KFAC performs like standard first-order optimizers like SGD or Adam. Thus, if we want to understand KFAC's effectiveness, we have to account for its damping strategy. This is achieved by our new explanation and even if one were to disagree with it, this finding deserves further attention and an explanation.

Ignoring the second Kronecker factor $\mathbf{E}_F\mathbf{E}_F^T$ of the approximate Fisher

A third important finding is that KFAC performs well, and often better,

5 We are aware of only one comparison, which unfortunately has serious limitations, see Supplementary Material 5.7.2.2.

without its second Kronecker factor. The fact that algorithms that are similar to KFAC without the second factor perform exceptionally well is consistent with prior work [42, 43, 190]. Moreover, Desjardins *et al.* [42] explicitly state that their algorithm performs more stably without the second Kronecker factor, which further confirms our findings. We emphasise that, without the second Kronecker factor, the preconditioning matrix of KFAC is independent of the loss (or derivatives of it), and thus cannot be seen as a classical second-order method. From a second-order viewpoint, dropping dependence on the loss should have detrimental effects, inconsistent with results from the studies above as well as ours.

Architectures with Parameter Sharing
Finally, we point to another intriguing finding from Grosse & Martens [170]. For architectures with parameter sharing, like CNNs or RNNs, approximating the Fisher by a Kronecker product requires additional, sometimes complex assumptions, which are not always satisfied. Grosse & Martens [170] explicitly investigate one such assumption for CNNs, pointing out that it is violated in architectures with average- rather than max-pooling. Nevertheless, KFAC performs very well in such architectures [171, 173, 175, 189]. This suggests that KFAC works well independently of how closely it is related to the Fisher, which is a puzzling observation when viewing KFAC as a natural gradient method. Again, our new explanation resolves this issue, since KFAC still performs gradient descent on neurons.[6]

In summary, we have shown that viewing KFAC as a second-order, natural gradient method is irreconcilable with a host of experimental results, from our as well as other studies. We then proposed a new, considerably improved explanation for KFAC's effectiveness. We also showed that the algorithm FOOF, which results from our explanation, can give further performance improvements compared to the state-of-the art optimizer KFAC.

5.7 SUPPLEMENTARY MATERIAL

5.7.1 *Kronecker-Factored Curvature Approximations for Laplace Posteriors*

A Kronecker factored approximation of the curvature has also been used in the context of Laplace posteriors [179] and this has been applied to continual learning [178]. In both context, empirical results are very encouraging.

6 In eq (5.4), we now impose one constraint per datapoint and per "location" at which the parameters are applied.

Our finding that, in the context of optimization, the effectiveness of KFAC does not rely on its similarity to the Fisher raises the question whether these other applications [178, 179] of Kronecker-factorisations of the curvature rely on proximity to the curvature matrix.

We point out that the applications from [178, 179] do not seem to rely on heuristic damping. Also in light of our findings, it remains plausible that without heuristic damping, KFAC is very similar to the Fisher. In other words, the below is a hypothesis, not a certainty. To evaluate it, it would be interesting to compare the performances of KFAC to Full Laplace as well as to the algorithm suggested below.

For simplicity of notation, we assume that the network has only one layer, but the analysis straightforwardly generalises to more layers. Suppose $\mathbf{W}_0$ is a local minimum of the negative log-likelihood of the parameters. Further denote the approximation to the posterior covariance by $\mathbf{\Sigma}$. For an approximate posterior to be effective, we require that parameters which are assigned high likelihood by the posterior actually do have high likelihood according to the data distribution. In other words, when a weight pertubation $\mathbf{V}$ satisfies that $\operatorname{vec}(\mathbf{V})^T \mathbf{\Sigma} \operatorname{vec}(\mathbf{V})$ is small (high likelihood according according to the approximate posterior), then the parameter $\mathbf{W}_0 + \mathbf{V}$ should have low loss (i.e. high likelihood according to the data).

For the Kronecker-factorisation, the first factor again is given by $\mathbf{A}\mathbf{A}^T$. Let us assume again that the second factor is dominated by a damping term (a very similar argument works if the second factor is predominantly diagonal), so that the posterior covariance is approximately $\mathbf{\Sigma} \approx \mathbf{A}\mathbf{A}^T \otimes \mathbf{I}$. Then, some easily verified calculations give

$$\operatorname{vec}(\mathbf{V})^T \mathbf{\Sigma} \operatorname{vec}(\mathbf{V}) \approx \operatorname{vec}(\mathbf{V})^T (\mathbf{A}\mathbf{A}^T \otimes \mathbf{I}) \operatorname{vec}(\mathbf{V}) = \sum_i \mathbf{v}_i^T (\mathbf{A}\mathbf{A}^T) \mathbf{v}_i \quad (5.6)$$

where $\mathbf{v}_i$ is the i-th row of $\mathbf{V}$, i.e. the set of weights connected to the i-th output neuron.[7] This expression being small means that each row of the pertubation $\mathbf{V}$ is near orthogonal to the input activations $\mathbf{A}$ (or more formally, it aligns with singular vectors of $\mathbf{A}$ which correspond to small singular values). This means that the layer's output, and consequently the networks output, get perturbed very little. This in turn means, that $\mathbf{W}_0 + \mathbf{V}$ has high-likelihood.

A simple test of this hypothesis would be to keep only the first kronecker-factor $\mathbf{A}\mathbf{A}^T$, replace the second one by the identity and check if the method

7 If the second kronecker-factor is not the identity, then there are additional cross terms of the form $b_{ij} \mathbf{v}_i^T (\mathbf{A}\mathbf{A}^T) \mathbf{v}_j$, where b_{ij} is the i, j-th entry of the second kronecker-factor.

performs equally well or better. Further, it would be interesting to compare the performance of kronecker-factored posterior to a full-laplace posterior (controlling for the amount of data given to both) and check if – analogous to our results for optimization – the kronecker-factored posterior outperforms the exact laplace posterior.

In fact, a very similar algorithm has already been developed independently [195]. It shows strong performance, indirectly supporting our hypothesis.

5.7.2 *Related Work*

We review generally related work as well as more specifically algorithms with similar updates rules to FOOF.

5.7.2.1 *Generally Related Work*

Natural gradients were proposed by Amari and colleagues, see e.g. [50] and its original motivation stems from information geometry [47]. It is closely linked to classical second-order optimization through the link of the Fisher to the Hessian and the Generalised Gauss Newton matrix [45, 49]. Moreover, natural gradients can be seen as a special case of Kalman filtering [196]. Interestingly, different filtering equations can be used to justify Adam's [51] update rule [52], see also [62].

There is a long history of approximating natural gradients and second order methods. For example, HF [39] exploits that Hessian-vector products are efficiently computable and uses the conjugate-gradient method to approximate products of the inverse Hessian and vectors. In this case, similarly to our application, the Hessian is usually subsampled, i.e. evaluated on a mini-batch. Other approximations of natural gradients include [39, 42, 197–201].

The intrinsic low rank structure of the (empirical) Fisher has been exploited in a number of setups by a number of papers including [53, 177, 202, 203].

Kronecker-factored approximations [41, 200] have become the basis of several optimization algorithms [172, 173, 176, 177]. Our contribution may shed light on why this is the case.

Moreover, Kronecker-factored approximation of the curvature can be used in the context of Laplace Posteriors [179], which can also be applied to continual learning [178]. A more detailed discussion of how this relates to our findings can be found in Section 5.7.1.

KFAC faces the problem of approximating a sum of kronecker-products by a single kronecker-product. This problem also occurs when approximating real time recurrent learning of recurrent networks [204–207] and in this context Benzing *et al.* [207] show how to obtain optimal biased and unbiased approximations. Our results suggest that it is not promising to apply these techniques to approximate natural gradients more accurately.

As briefly mentioned, FOOF is related to the idea of optimizing modules of a nested function independently, e.g. [191–194].

It may also be worth noting that FOOF is evocative of target propagation [208, 209], but we are not aware of a formal link between these methods.

5.7.2.2 *Comparison between HF and KFAC*

The subsampled natural gradient method upon which many of our results rely was first described in [40]. On top of their useful, important theoretical results, they also provide an empirical evaluation of their method, and – to the best of our knowledge – the only published comparison of KFAC and HF.

Unfortunately, there is very strong evidence that all methods considered there are heavily undertuned or that there is another issue. To see this, note that in [40] a network trained on MNIST with one hidden layer of 500 neurons achieves a training loss of around 0.3 and a test accuracy of less than 95% for all considered optimizers. This is much worse than standard results and clearly not representative of normal neural network training. We quickly verified that in exactly the same setting, with KFAC we are able to obtain a loss which is more than 100x smaller and a test accuracy of 98%.[8]

5.7.2.3 *Theoretical Work*

There also is a large body of work on theoretical convergence properties of Natural Gradients. We give a brief, incomplete overview here and refer to [210] for a more thorough discussion.

[176] analyse the convergence of natural gradients in linear networks. Interestingly, they show that for linear networks applied to regression

8 We used the same network, same activation function and same number of epochs. Weight initialisation scheme, batch size and preprocessing were not described in the original experiments, so we used batch size 100 and the same initialisation and preprocessing as in our other MNIST experiments (which has no data augmentation).

problems (with homoscedastic noise), inverting a block-diagonal, Kronecker-Factored approximation of the curvature results in exact natural gradients. We point out that the empirical results in non-linear networks from [176] essentially amount to a re-discovery of KFAC, as such they do not contradict our results. In particular, they are also based on heuristic damping.

For non-linear, strongly overparametrised two-layer networks in which only the first layer is trained, [210] recently gave a convergence analysis of both natural gradients and KFAC. Note that [210] do not establish similarity between KFAC and Natural Gradients but rather give two separate convergence proofs.

Both these theoretical results [176, 181] do not account for any form of damping, so they have to be seen as independent of the empirically well-performing version of KFAC and our investigation.

A set of interesting theoretical results by [211] shows that the Fisher Information in deep neural networks has a pathological spectrum – in particular, they show that the Fisher is flat in most directions. This view may well give a theoretical intuition for why Subsampled Natural Gradients do often not notably outperform SGD.

5.7.2.4 *Related Work from Bayesian ML*

Similar to our new view on optimization is [195], which is a Bayesian Posterior approximation and can (roughly) be viewed as considering distributions over neuron activations rather than in weight space directly, similarly to how FOOF performs optimization steps on neuron activations rather than on weights directly.

5.7.2.5 *Algorithms with similar update rules*

While it is not immediately visible due to a re-parametrisation employed in [42], Natural Neural Networks [42] (NNN) propose a mathematically very similar update rule to FOOF (and KFAC). Unlike FOOF, NNN centers layer inputs by subtracting the mean activation (or an estimate thereof), but like FOOF they ignore the second kronecker factor of KFAC.

Like KFAC, NNN is derived as a block-diagonal, kronecker-factored approximation of the Fisher. As we already pointed out, this is very puzzling, since NNN approximates the Fisher by a zero-th order matrix, ignoring all first- and second-order information. In this sense, and bearing in mind our previous experiments, NNN should not be seen as a natural gradi-

ent method and our results offer an explanation why it is nevertheless so effective.

From an implementational viewpoint, FOOF is preferable to NNN mainly because it requires inverting matrices rather than computing SVDs. In practice computing inverses is both considerably faster (a factor of 10 or so as found in some quick experiments) and more stable then computing the SVD, as NNN does (SVD algorithms don't always converge).

With yet another context and motivation, [43] also proposes a similar update rule focussing on full-batch descent. The motivation can be roughly rephrased as imposing proximity constraints on neuron activations. Very recently, their motivation and algorithm seems to have been re-described in [190] without noting this link. In particular, the derivation of [43] gives an alternative perspective on the update equation of FOOF.

Among other differences, [43, 190] (1) seem not to discuss unbiased stochastic versions of their algorithms, (2) seem less computationally efficient: results in [43] fall short of adam in terms of wall-clock time and [190] does not provide direct wall-clock time comparisons with standard first-order optimizers, (3) only discuss fully-connected architectures (4) do not perform investigations into the connection of KFAC to natural gradients or first-order methods.

Note also that the framework from [196] can be applied to interpret FOOF as applying Kalman filtering to each layer individually. Thus, in some vague sense, FOOF is bayes-optimal and some may find this to be an enticing explanation for FOOF's strong empirical performance.

5.7.3 *Additional Experiments*

Here we present additional experiments, lines correspond to the average across three seeds.

In Section 5.7.3.2, there are experiments analogous to Figures 5.1-5.4 from the main chapter, but on MNIST rather than Fashion-MNIST. Results are in line with the ones on Fashion-MNIST, but effects are usually smaller, presumably due to MNIST-classification being a very simple task for MLPs.

In Section 5.7.3.3, there are comparisons of subsampled natural gradients to subsampled KFAC for a ResNet on CIFAR10 and for autoencoder experiments. The fact that KFAC performs considerably better than natural gradients strongly suggests that also in these settings KFAC cannot be seen as a natural gradient method.

In Section 5.7.3.4, we show that heuristic damping is crucial for performance of KFAC for a ResNet on CIFAR10 and for autoencoder experiments. This further supports the claim that KFAC should not be seen as a natural gradient method and suggests that similarity to FOOF is important for KFAC.

In Section 5.7.3.5, we show performance comparisons between KFAC and FOOF for different benchmarks and architectures. Figure 5.14 contains performance of a Wide ResNet18 on CIFAR 100 (rather than CIFAR 10 in the main chapter). Figures 5.15, 5.16 contain training data for a VGG11 network on SVHN and additionally show how different algortihms are affected by different batch sizes.

5.7.3.1 *Limitations of our Explanation in Auto-Encoder Settings*

Figures 5.17-5.19 contain autoencoder experiments on MNIST, Curves and Faces. Here, we make a somewhat puzzling observation: When we follow the KFAC training trajectory, FOOF makes more progress per parameter udpate than KFAC. Nevertheless, when we use FOOF for training (and follow the FOOF trajectory), we obtain slightly worse results than for KFAC. This may suggest that in the autoencoder experiments KFAC chooses a different trajectory that is easier to optimize than FOOF.

This suggests that our explanation of KFACs performance, while capturing many key-characteristics of KFAC, has some limitations.

We re-emphasise that similarity to FOOF remains a significantly better explanation for KFAC's performance than similarity to natural gradients, also in the autoencoder setting (recall Figure 5.11).

As an additional experiment we run KFAC with the empirical Fisher, rather than an MC approximation. Despite its name, the empirical Fisher is usually argued to be a poor approximation of the Fisher [45, 46]. Nevertheless we find that KFAC works equally well with the empirical Fisher, see Figure 5.20, supporting the view that KFAC's effectiveness is not directly linked to the Fisher and hinting at the fact that a full, principled explanation of KFAC's slight advantage over FOOF in the autoencoder setting may be difficult to come by.

5.7.3.2 *Experiments anaologous to main chapter but on MNIST rather than Fashion MNIST*

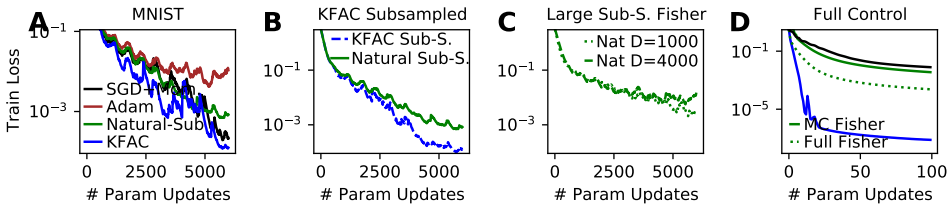


FIGURE 5.6: **Paradoxically, KFAC – an approximate second-order method – outperforms exact second-order udpates in standard as well as important control settings.** Same as Figure 5.1 but on MNIST rather than Fashion MNIST.

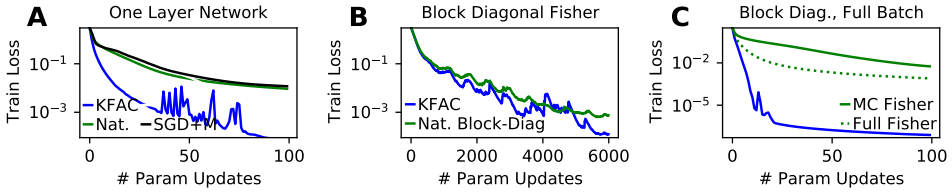


FIGURE 5.7: **Advantage of KFAC over exact, subsampled Natural Gradients is not due to block-diagonal structure.** Same as Figure 5.2 but on MNIST rather than Fashion MNIST.

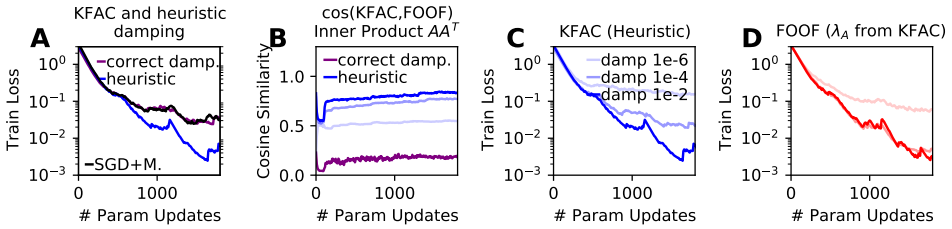


FIGURE 5.8: **Heuristic Damping increases KFAC's performance as well as its similarity to first-order method FOOF.** Same as Figure 5.3 but on MNIST rather than Fashion MNIST.

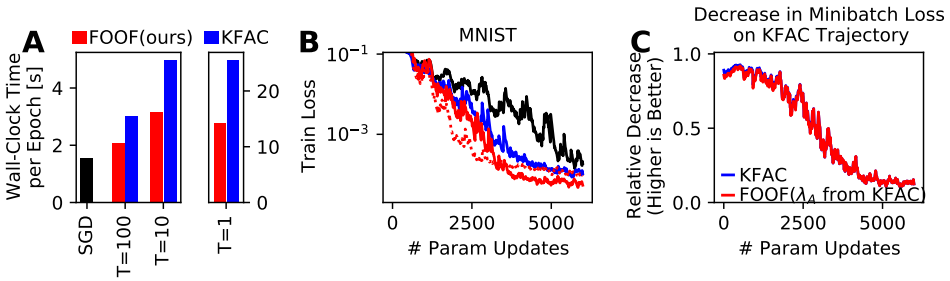


FIGURE 5.9: **FOOF outperforms KFAC in terms of both per-update progress and computation cost.** Same as Figure 5.4 but on MNIST rather than Fashion MNIST.

5.7.3.3 *Subsampled Natural Gradients vs Subsampled KFAC*

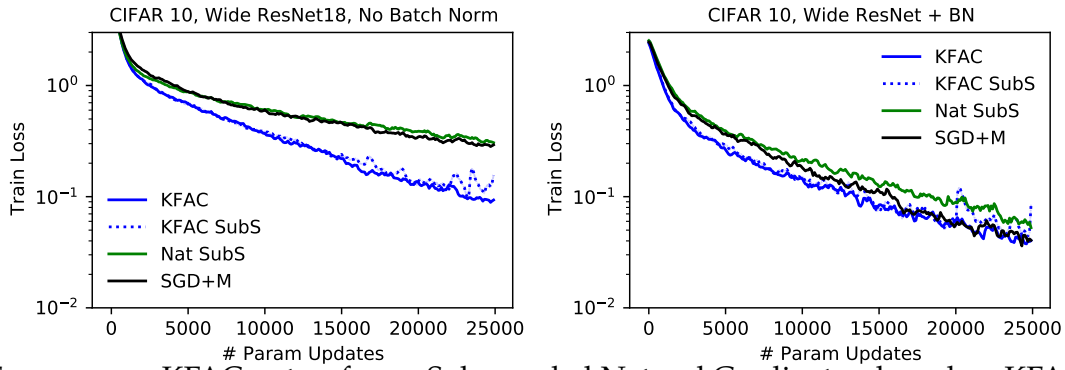


FIGURE 5.10: KFAC outperforms Subsampled Natural Gradients, also when KFAC is subsampled and uses exactly the same amount of data as Natural gradients to estimate the curvature. This is analogous to Figure 5.1B, but on ConvNets and with a more complicated dataset. It confirms our claim that KFAC does not rely on second-order information. Note that with large damping, natural gradients becomes approximately equal to SGD – thus the difference seen between SGD+M and natural gradients is due to momentum.

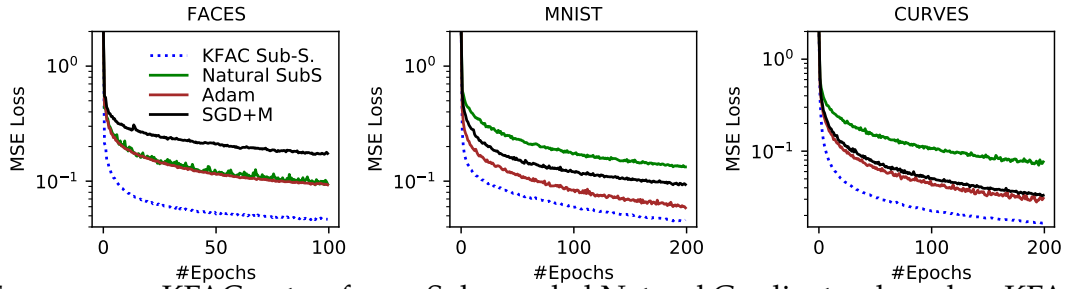


FIGURE 5.11: KFAC outperforms Subsampled Natural Gradients, also when KFAC is subsampled and uses exactly the same amount of data as Natural gradients to estimate the curvature. Autoencoder Experiments. We confirmed that Natural Gradients do not perform worse than SGD without momentum. In other words the advantage of SGD+M vs Natural Gradients on MNIST and Curves is due to using momentum.

5.7.3.4 *Effect of Heuristic Damping on KFAC*

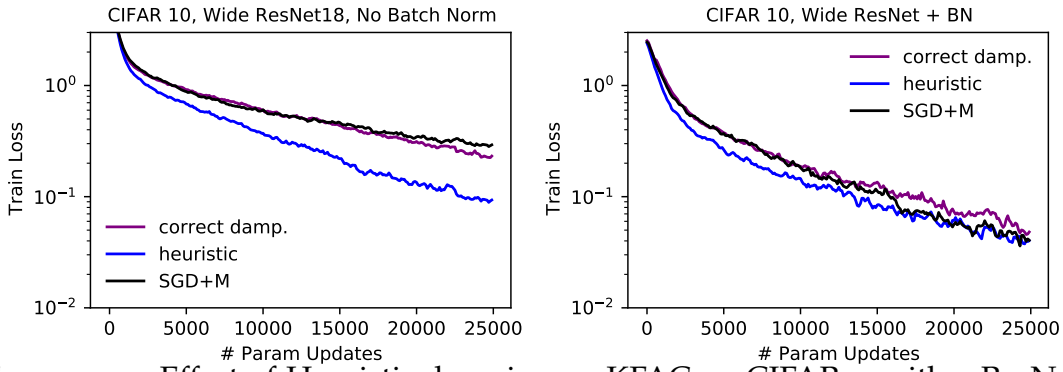


FIGURE 5.12: Effect of Heuristic damping on KFAC on CIFAR10 with a ResNet. Analogously to Figure 5.3A, we find that heuristic damping is essential for KFAC's performance.

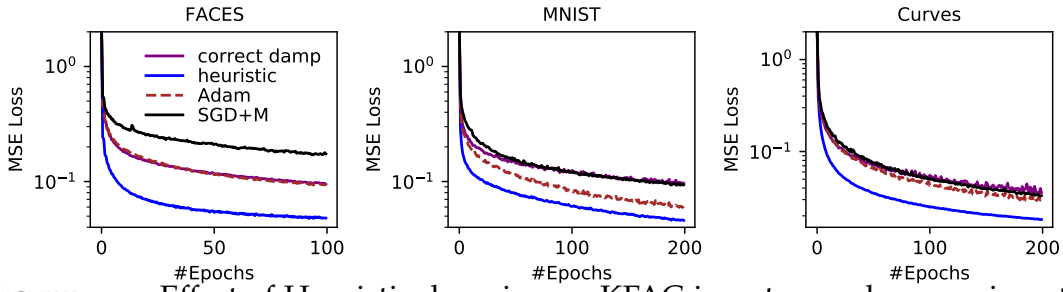


FIGURE 5.13: Effect of Heuristic damping on KFAC in autoencoder experiments. Analogously to Figure 5.3A, we find that heuristic damping is essential for KFAC's performance.

5.7.3.5 *Performance Comparisons*

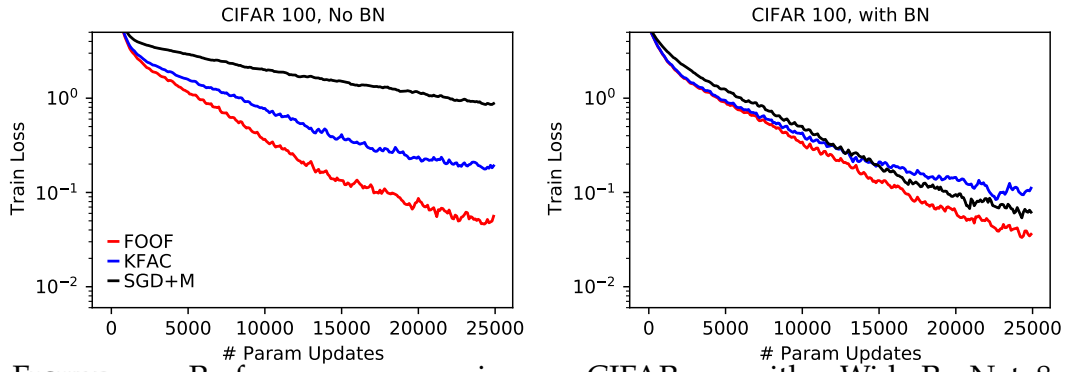


FIGURE 5.14: Performance comparison on CIFAR 100 with a Wide ResNet18.

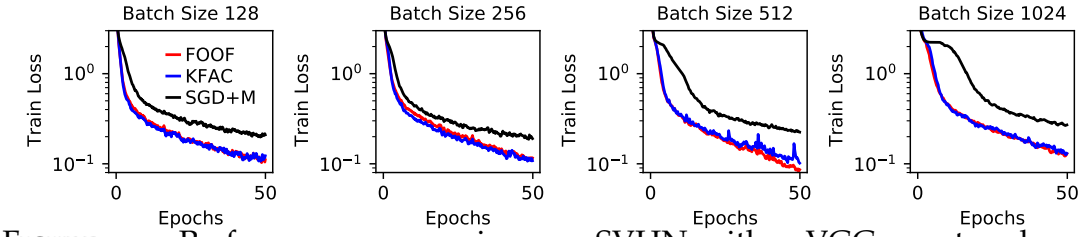


FIGURE 5.15: Performance comparison on SVHN with a VGG11 network and different batch sizes. See also Figure 5.16 for same data portrayed differently.

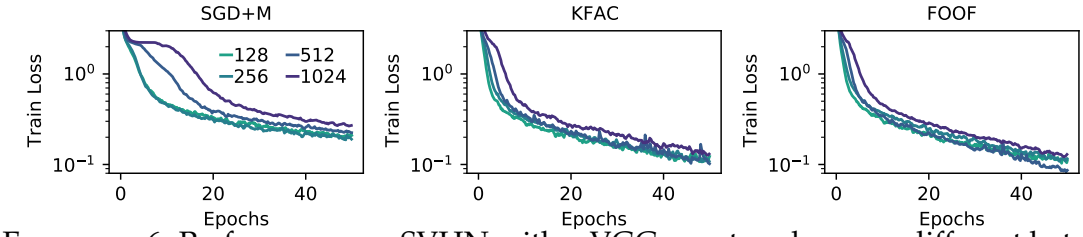


FIGURE 5.16: Performance on SVHN with a VGG11 network across different batch sizes for different algorithms. See also Figure 5.15 for same data portrayed differently. Note that colour coding differs from remaining plots in this chapter.

5.7.3.6 *Pseudocode, Implementation, Hyperparameters*

Pseudocode for FOOF (including amortisation techniques) is given in Algorithm 1. Notation is anaologous to Section 5.2.

Initialisation: One detail omitted in the pseudocode is initialisation of $\boldsymbol{\Sigma}$ and $\mathbf{P}$. There is different ways to do this. We decided to perform Line 17

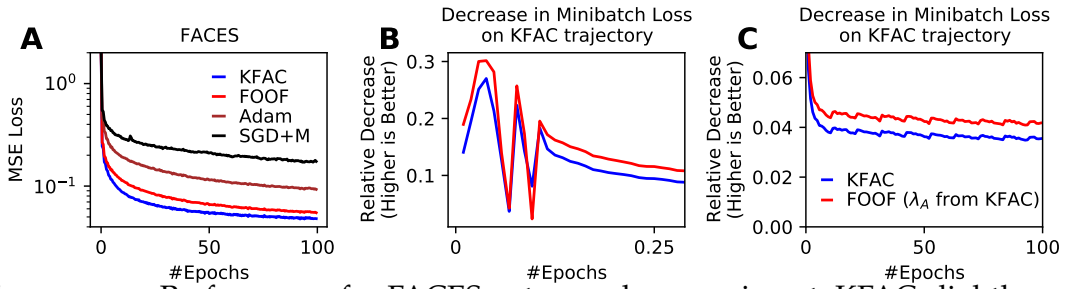


FIGURE 5.17: Performance for FACES autoencoder experiment. KFAC slightly outperforms FOOF, but when FOOF is on KFAC trajectory it typically makes more progress per udpate. This may suggest that the advantage of KFAC is due to choosing a different optimization trajectory. (B) shows same data as (C) with a different axes zoom.

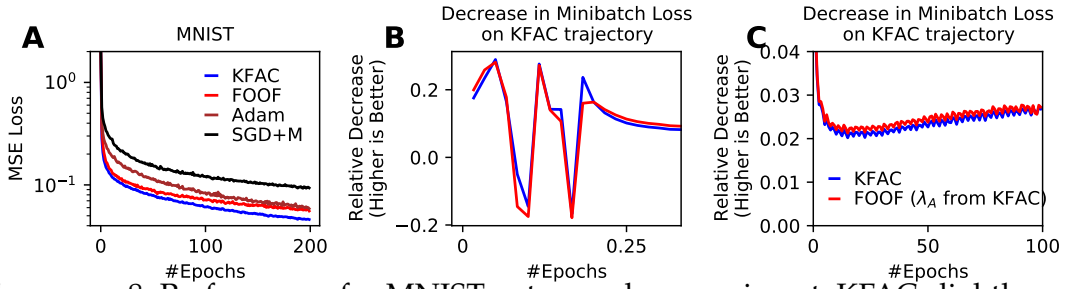


FIGURE 5.18: Performance for MNIST autoencoder experiment. KFAC slightly outperforms FOOF, but when FOOF is on KFAC trajectory it typically makes more progress per udpate. This may suggest that the advantage of KFAC is due to choosing a different optimization trajectory. (B) shows same data as (C) with a different axes zoom.

of Algorithm 1 for a number of minibatches (50) before training and then executing line Line 10 once. In addition, we make sure the exponentially moving average is normalised.

Amortisation Choices: Amortising the overhead of FOOF is achieved by choosing S, T suitably (large T and small S give the best runtimes). For fully connected layers updating $\mathbf{\Sigma}$ is cheap and we choose $S = T$ (i.e. $\mathbf{\Sigma}$ is updated at every step), we reported results for $T = 1$ and $T = 100$. For the ResNet, computing $\mathbf{A}\mathbf{A}^T$ is more expensive and we chose $T = 500$ (one inversion per epoch) and $S = 10$. Additional experiments (not shown) suggest that $\mathbf{\Sigma}$ can be estimated robustly on few datapoints and that it changes slowly during training.

Hyperparameter Choices: Note that a discussion of hyperparameter robustness is also provided in Appendix D of Benzing [44]. We chose $m = 0.95$ following [41], brief experiments with $m = 0.999$ seemed to give very similar results. For damping λ and learning rate η, we performed

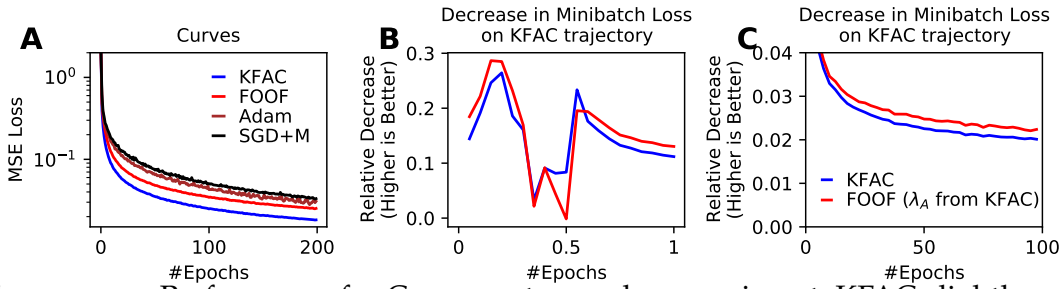


FIGURE 5.19: Performance for Curves autoencoder experiment. KFAC slightly outperforms FOOF, but when FOOF is on KFAC trajectory it typically makes more progress per udpate. This may suggest that the advantage of KFAC is due to choosing a different optimization trajectory. (B) shows same data as (C) with a different axes zoom.

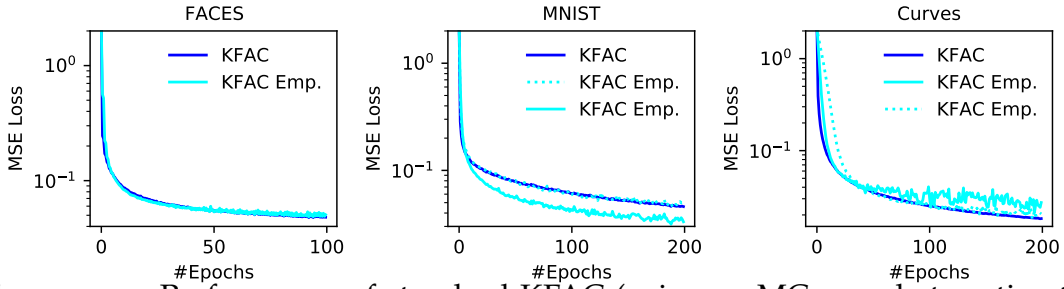


FIGURE 5.20: Performance of standard KFAC (using an MC sample to estimate the Fisher) and a version of KFAC using the empirical Fisher. Solid an dashed cyan lines show different hyperparametrisations of the same algorithm. The advantage fo the empirical Fisher on MNIST seems to be due to allowing different hyperparametrisations to be stable.

grid searches. It may be interesting that a bayesian interpretation of FOOF (details omitted) suggests choosing λ as the precision used for standard weight initialisation schemes (e.g. Kaiming Normal initialisation) and seems to work well. If we choose this λ, we only need to tune the learning rate of FOOF, so that the required tuning is analogous to that of SGD. Alternatively, we typically fount $\lambda = 100$ to work well, but the exact magnitude may depend on implementation details (in particular, how factors are scaled and how they depend on the batch size).

Implementation and Convolutional Layers: A PyTorch implementation of FOOF using for- and backward hooks is simple. The implementation, in particular computing $\mathbf{A}\mathbf{A}^T$, is most straightforward for fully connected layers, but can be extended to layers with parameter sharing. For example in CNNs, we can interpret the convolution as a standard matrix multiplication by "extracting/unfolding" individual patches (see e.g. [170]) and then

proceed as before. This is what our implementation does. A more efficient technique avoiding explicitly extracting patches (at the cost of making small approximations) is presented in [195].

Algorithm 1 Gradient Descent on Neurons (FOOF)

Hyperparameters: learning rate η, damping strength λ, exponential decay factor m, inversion period T, number of updates for input covariance $S \leq T$

Initialise: $t = 0$; For each layer ℓ: Weights $\mathbf{W}_\ell$ (e.g. Kaiming-He init), exponential average $\mathbf{\Sigma}_\ell$ of $\mathbf{A}_\ell \mathbf{A}_\ell^T$ and its damped inverse $\mathbf{P}_\ell = (\mathbf{\Sigma}_\ell + \lambda \mathbf{I})^{-1}$ (see Supplementary Material 5.7.3.6 for details)

while train **do**
  Perform Standard Forward and Backward Pass
    For Current Mini-Batch With Loss L
  for each layer ℓ **do**
    $\mathbf{W}_\ell \leftarrow \mathbf{W}_\ell - \eta \mathbf{P}_\ell \nabla_{\mathbf{W}_\ell} L$ ▷ Update Parameters as in Eq. 5.5
    if $(t \bmod T) == 0$ **then**
      $\mathbf{P}_\ell \leftarrow (\mathbf{\Sigma}_\ell + \lambda \mathbf{I})^{-1}$ ▷ Update Damped Inverse of Moving of Average $\mathbf{A}_\ell \mathbf{A}_\ell^T$ every T steps
    end if
    if $((t + S) \bmod T) \in \{0, \ldots, S-1\}$ **then**
      $\mathbf{\Sigma} \leftarrow m \cdot \mathbf{\Sigma}_\ell + (1 - m) \cdot \mathbf{A}_\ell \mathbf{A}_\ell^T$ ▷ Update Moving Average of $\mathbf{A}_\ell \mathbf{A}_\ell^T$ beginning S steps before inversion in line 10. $\mathbf{A}_\ell$ is defined as in Section 5.2.
    end if
  end for
  $t \leftarrow t + 1$
end while

BIBLIOGRAPHY

1. Chen, T., Kornblith, S., Norouzi, M. & Hinton, G. *A simple framework for contrastive learning of visual representations* in *International conference on machine learning* (2020), 1597.
2. Zbontar, J., Jing, L., Misra, I., LeCun, Y. & Deny, S. *Barlow twins: Self-supervised learning via redundancy reduction* in *International Conference on Machine Learning* (2021), 12310.
3. Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., *et al. Learning transferable visual models from natural language supervision* in *International Conference on Machine Learning* (2021), 8748.
4. Blum, A. L. & Rivest, R. L. Training a 3-node neural network is NP-complete. *Neural Networks* **5**, 117 (1992).
5. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *nature* **323**, 533 (1986).
6. Geiping, J., Goldblum, M., Pope, P. E., Moeller, M. & Goldstein, T. Stochastic training is not necessary for generalization. *arXiv preprint arXiv:2109.14119* (2021).
7. Sutskever, I., Martens, J., Dahl, G. & Hinton, G. *On the importance of initialization and momentum in deep learning* in *International conference on machine learning* (2013), 1139.
8. Choi, D., Shallue, C. J., Nado, Z., Lee, J., Maddison, C. J. & Dahl, G. E. On empirical comparisons of optimizers for deep learning. *arXiv preprint arXiv:1910.05446* (2019).
9. French, R. M. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences* **3**, 128 (1999).
10. Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A. & Bengio, Y. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211* (2013).
11. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., *et al.* Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* **114**, 3521 (2017).

12. Rebuffi, S.-A., Kolesnikov, A., Sperl, G. & Lampert, C. H. *icarl: Incremental classifier and representation learning* in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (2017), 2001.
13. Zenke, F., Poole, B. & Ganguli, S. *Continual learning through synaptic intelligence* in *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), 3987.
14. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M. & Tuytelaars, T. *Memory aware synapses: Learning what (not) to forget* in *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), 139.
15. Huszár, F. Note on the quadratic penalties in elastic weight consolidation. *Proceedings of the National Academy of Sciences*, 201717042 (2018).
16. Ritter, H., Botev, A. & Barber, D. *Online structured laplace approximations for overcoming catastrophic forgetting* in *Advances in Neural Information Processing Systems* (2018), 3738.
17. Chaudhry, A., Dokania, P. K., Ajanthan, T. & Torr, P. H. *Riemannian walk for incremental learning: Understanding forgetting and intransigence* in *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), 532.
18. Yin, D., Farajtabar, M. & Li, A. *SOLA: Continual Learning with Second-Order Loss Approximation* 2020.
19. Schwarz, J., Luketina, J., Czarnecki, W. M., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R. & Hadsell, R. Progress & compress: A scalable framework for continual learning. *arXiv preprint arXiv:1805.06370* (2018).
20. Liu, X., Masana, M., Herranz, L., Van de Weijer, J., Lopez, A. M. & Bagdanov, A. D. *Rotate your networks: Better weight consolidation and less catastrophic forgetting* in *2018 24th International Conference on Pattern Recognition (ICPR)* (2018), 2262.
21. Park, D., Hong, S., Han, B. & Lee, K. M. *Continual learning by asymmetric loss approximation with single-side overestimation* in *Proceedings of the IEEE International Conference on Computer Vision* (2019), 3335.
22. Lee, J., Hong, H. G., Joo, D. & Kim, J. *Continual Learning with Extended Kronecker-factored Approximate Curvature* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), 9001.
23. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural computation* **9**, 1735 (1997).

24. Glorot, X. & Bengio, Y. *Understanding the difficulty of training deep feedforward neural networks* in *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (2010), 249.
25. He, K., Zhang, X., Ren, S. & Sun, J. *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification* in *Proceedings of the IEEE international conference on computer vision* (2015), 1026.
26. Ioffe, S. & Szegedy, C. *Batch normalization: Accelerating deep network training by reducing internal covariate shift* in *International conference on machine learning* (2015), 448.
27. Ba, J. L., Kiros, J. R. & Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).
28. He, K., Zhang, X., Ren, S. & Sun, J. *Deep residual learning for image recognition* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), 770.
29. De, S. & Smith, S. Batch normalization biases residual blocks towards the identity function in deep networks. *Advances in Neural Information Processing Systems* **33**, 19964 (2020).
30. Hochreiter, S. Untersuchungen zu dynamischen neuronalen Netzen. *Diploma, Technische Universität München* **91** (1991).
31. Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., *et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies* 2001.
32. Li, H., Xu, Z., Taylor, G., Studer, C. & Goldstein, T. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems* **31** (2018).
33. Orhan, A. E. & Pitkow, X. Skip connections eliminate singularities. *arXiv preprint arXiv:1701.09175* (2017).
34. Martens, J., Ballard, A., Desjardins, G., Swirszcz, G., Dalibard, V., Sohl-Dickstein, J. & Schoenholz, S. S. Rapid training of deep neural networks without skip connections or normalization layers using deep kernel shaping. *arXiv preprint arXiv:2110.01765* (2021).
35. Zhang, G., Botev, A. & Martens, J. Deep Learning without Shortcuts: Shaping the Kernel with Tailored Rectifiers. *arXiv preprint arXiv:2203.08120* (2022).
36. Brock, A., De, S., Smith, S. L. & Simonyan, K. *High-performance large-scale image recognition without normalization* in *International Conference on Machine Learning* (2021), 1059.

37. Patterson, D., Gonzalez, J., Hölzle, U., Le, Q., Liang, C., Munguia, L.-M., Rothchild, D., So, D., Texier, M. & Dean, J. The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink. *arXiv preprint arXiv:2204.05149* (2022).
38. Dayma, B. & Anil, R. *Evaluation of Distributed Shampoo* [Online; accessed 21-June-2022]. 2022.
39. Martens, J. *et al. Deep learning via hessian-free optimization.* in *ICML* **27** (2010), 735.
40. Ren, Y. & Goldfarb, D. Efficient subsampled gauss-newton and natural gradient methods for training neural networks. *arXiv preprint arXiv:1906.02353* (2019).
41. Martens, J. & Grosse, R. *Optimizing neural networks with kronecker-factored approximate curvature* in *International conference on machine learning* (2015), 2408.
42. Desjardins, G., Simonyan, K., Pascanu, R. & Kavukcuoglu, K. Natural neural networks. *arXiv preprint arXiv:1507.00210* (2015).
43. Frerix, T., Möllenhoff, T., Moeller, M. & Cremers, D. Proximal backpropagation. *arXiv preprint arXiv:1706.04638* (2017).
44. Benzing, F. Gradient Descent on Neurons and its Link to Approximate Second-Order Optimization. *arXiv preprint arXiv:2201.12250* (2022).
45. Martens, J. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193* (2014).
46. Kunstner, F., Balles, L. & Hennig, P. Limitations of the empirical Fisher approximation for natural gradient descent. *arXiv preprint arXiv:1905.12558* (2019).
47. Amari, S.-i. & Nagaoka, H. *Methods of information geometry* (American Mathematical Soc., 2000).
48. Heskes, T. On "natural" learning and pruning in multilayered perceptrons. *Neural Computation* **12**, 881 (2000).
49. Pascanu, R. & Bengio, Y. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584* (2013).
50. Amari, S.-I. Natural gradient works efficiently in learning. *Neural computation* **10**, 251 (1998).
51. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

52. Aitchison, L. Bayesian filtering unifies adaptive and non-adaptive neural network optimization methods. *arXiv preprint arXiv:1807.07540* (2018).

53. Agarwal, N., Bullins, B., Chen, X., Hazan, E., Singh, K., Zhang, C. & Zhang, Y. *Efficient full-matrix adaptive regularization* in *International Conference on Machine Learning* (2019), 102.

54. Benzing, F. *Unifying Importance Based Regularisation Methods for Continual Learning* in *International Conference on Artificial Intelligence and Statistics* (2022), 2372.

55. Van de Ven, G. M. & Tolias, A. S. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734* (2019).

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

57. Aljundi, R., Kelchtermans, K. & Tuytelaars, T. *Task-free continual learning* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), 11254.

58. Lan, J., Liu, R., Zhou, H. & Yosinski, J. *LCA: Loss change allocation for neural network training* in *Advances in Neural Information Processing Systems* (2019), 3614.

59. Li, Z. & Hoiem, D. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence* **40**, 2935 (2017).

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

61. Loo, N., Swaroop, S. & Turner, R. E. Generalized Variational Continual Learning. *arXiv preprint arXiv:2011.12328* (2020).

62. Khan, M., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y. & Srivastava, A. *Fast and scalable bayesian deep learning by weight-perturbation in adam* in *International Conference on Machine Learning* (2018), 2611.

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

64. Jastrzebski, S., Kenton, Z., Ballas, N., Fischer, A., Bengio, Y. & Storkey, A. On the relation between the sharpest directions of DNN loss and the SGD step length. *arXiv preprint arXiv:1807.05031* (2018).

65. Mirzadeh, S. I., Farajtabar, M., Pascanu, R. & Ghasemzadeh, H. *Understanding the Role of Training Regimes in Continual Learning* 2020.

66. keras. *Official Keras CNN example* `https://github.com/keras-team/keras/blob/master/examples/cifar10_cnn.py`. Accessed: 2020-09-25.

67. Swaroop, S., Nguyen, C. V., Bui, T. D. & Turner, R. E. Improving and understanding variational continual learning. *arXiv preprint arXiv:1905.02099* (2019).

68. McCloskey, M. & Cohen, N. J. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. *Psychology of Learning and Motivation - Advances in Research and Theory* **24**, 109 (1989).

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

70. Srivastava, R. K., Masci, J., Kazerounian, S., Gomez, F. & Schmidhuber, J. *Compete to Compute* in *Advances in Neural Information Processing Systems 26* (eds Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z. & Weinberger, K. Q.) (Curran Associates, Inc., 2013), 2310.

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

72. Lopez-Paz, D. & Ranzato, M. *Gradient episodic memory for continual learning* in *Advances in Neural Information Processing Systems* (2017), 6467.

73. Shin, H., Lee, J. K., Kim, J. & Kim, J. *Continual learning with deep generative replay* in *Advances in Neural Information Processing Systems* (2017), 2990.

74. Kemker, R. & Kanan, C. Fearnet: Brain-inspired model for incremental learning. *arXiv preprint arXiv:1711.10563* (2017).

75. Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H. & Ranzato, M. Continual learning with tiny episodic memories. *arXiv preprint arXiv:1902.10486* (2019).

76. 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. *arXiv preprint arXiv:1701.08734* (2017).

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

78. Golkar, S., Kagan, M. & Cho, K. Continual learning via neural pruning. *arXiv preprint arXiv:1903.04476* (2019).

79. Von Oswald, J., Henning, C., Sacramento, J. & Grewe, B. F. Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695* (2019).

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

81. Thomas, V., Pedregosa, F., Merriënboer, B., Manzagol, P.-A., Bengio, Y. & Le Roux, N. *On the interplay between noise and curvature and its effect on optimization and generalization* in *International Conference on Artificial Intelligence and Statistics* (2020), 3503.

82. Schug, S., Benzing, F. & Steger, A. Presynaptic stochasticity improves energy efficiency and helps alleviate the stability-plasticity dilemma. *Elife* **10**, e69884 (2021).

83. Del Castillo, J. & Katz, B. Quantal components of the end-plate potential. *The Journal of Physiology* **124**, 560 (1954).

84. Branco, T. & Staras, K. The probability of neurotransmitter release: variability and feedback control at single synapses. *Nature Reviews Neuroscience* **10**, 373 (2009).

85. Shannon, C. E. A Mathematical Theory of Communication. *Bell System Technical Journal* **27**, 379 (1948).

86. Faisal, A. A., White, J. A. & Laughlin, S. B. Ion-channel noise places limits on the miniaturization of the brain's wiring. *Current Biology* **15**, 1143 (2005).

87. Niven, J. E. & Laughlin, S. B. Energy limitation as a selective pressure on the evolution of sensory systems. *Journal of Experimental Biology* **211**, 1792 (2008).

88. Attwell, D. & Laughlin, S. B. An energy budget for signaling in the grey matter of the brain. *Journal of Cerebral Blood Flow & Metabolism* **21**, 1133 (2001).

89. Harris, J. J., Jolivet, R. & Attwell, D. Synaptic Energy Use and Supply. *Neuron* **75**, 762 (2012).

90. Llera-Montero, M., Sacramento, J. & Costa, R. P. Computational roles of plastic probabilistic synapses. *Current Opinion in Neurobiology* **54**. Neurobiology of Learning and Plasticity, 90 (2019).

91. Levy, W. B. & Baxter, R. A. Energy efficient neural codes. *Neural computation* **8**, 531 (1996).

92. Levy, W. B. & Baxter, R. A. Energy-efficient neuronal computation via quantal synaptic failures. *Journal of Neuroscience* **22**, 4746 (2002).

93. Sengupta, B., Laughlin, S. B. & Niven, J. E. Balanced excitatory and inhibitory synaptic currents promote efficient coding and metabolic efficiency. *PLoS Comput Biol* **9**, e1003263 (2013).

94. Savtchenko, L. P., Sylantyev, S. & Rusakov, D. A. Central synapses release a resource-efficient amount of glutamate. *Nature neuroscience* **16**, 10 (2013).

95. Harris, J. J., Jolivet, R., Engl, E. & Attwell, D. Energy-efficient information transfer by visual pathway synapses. *Current Biology* **25**, 3151 (2015).

96. Harris, J. J., Engl, E., Attwell, D. & Jolivet, R. B. Energy-efficient information transfer at thalamocortical synapses. *PLoS computational biology* **15**, e1007226 (2019).

97. Yang, G., Pan, F. & Gan, W.-B. Stably maintained dendritic spines are associated with lifelong memories. *Nature* **462**, 920 (2009).

98. Hayashi-Takagi, A., Yagishita, S., Nakamura, M., Shirai, F., Wu, Y. I., Loshbaugh, A. L., Kuhlman, B., Hahn, K. M. & Kasai, H. Labelling and optical erasure of synaptic memory traces in the motor cortex. *Nature* **525**, 333 (2015).

99. Hardingham, N. R., Read, J. C., Trevelyan, A. J., Nelson, J. C., Jack, J. J. B. & Bannister, N. J. Quantal analysis reveals a functional correlation between presynaptic and postsynaptic efficacy in excitatory connections from rat neocortex. *Journal of Neuroscience* **30**, 1441 (2010).

100. Sakamoto, H., Ariyoshi, T., Kimpara, N., Sugao, K., Taiko, I., Takikawa, K., Asanuma, D., Namiki, S. & Hirose, K. Synaptic weight set by Munc13-1 supramolecular assemblies. *Nature neuroscience* **21**, 41 (2018).

101. Kriegeskorte, N. Deep neural networks: a new framework for modeling biological vision and brain information processing. *Annual review of vision science* **1**, 417 (2015).

102. Yamins, D. L. & DiCarlo, J. J. Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience* **19**, 356 (2016).

103. Kell, A. J., Yamins, D. L., Shook, E. N., Norman-Haignere, S. V. & McDermott, J. H. A task-optimized neural network replicates human auditory behavior, predicts brain responses, and reveals a cortical processing hierarchy. *Neuron* **98**, 630 (2018).

104. Banino, A., Barry, C., Uria, B., Blundell, C., Lillicrap, T., Mirowski, P., Pritzel, A., Chadwick, M. J., Degris, T., Modayil, J., *et al.* Vector-based navigation using grid-like representations in artificial agents. *Nature* **557**, 429 (2018).

105. Cueva, C. J. & Wei, X.-X. Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. *arXiv preprint arXiv:1803.07770* (2018).

106. Mattar, M. G. & Daw, N. D. Prioritized memory access explains planning and hippocampal replay. *Nature neuroscience* **21**, 1609 (2018).

107. Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review* **65**, 386 (1958).

108. LeCun, Y. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/* (1998).

109. Cichon, J. & Gan, W.-B. Branch-specific dendritic Ca 2+ spikes cause persistent synaptic plasticity. *Nature* **520**, 180 (2015).

110. Zeno, C., Golan, I., Hoffer, E. & Soudry, D. Task agnostic continual learning using online variational Bayes. *arXiv preprint arXiv:1803.10123* (2018).

111. Branco, T., Staras, K., Darcy, K. J. & Goda, Y. Local dendritic activity sets release probability at hippocampal synapses. *Neuron* **59**, 475 (2008).

112. Larkman, A., Hannay, T., Stratford, K. & Jack, J. Presynaptic release probability influences the locus of long-term potentiation. *Nature* **360**, 70 (1992).

113. Lisman, J. & Raghavachari, S. A unified model of the presynaptic and postsynaptic changes during LTP at CA1 synapses. *Science's STKE* **2006**, re11 (2006).

114. Bayazitov, I. T., Richardson, R. J., Fricke, R. G. & Zakharenko, S. S. Slow presynaptic and fast postsynaptic components of compound long-term potentiation. *Journal of Neuroscience* **27**, 11510 (2007).

115. Sjöström, P. J., Turrigiano, G. G. & Nelson, S. B. Multiple forms of long-term plasticity at unitary neocortical layer 5 synapses. *Neuropharmacology* **52**, 176 (2007).

116. Bliss, T. V. & Collingridge, G. L. Expression of NMDA receptor-dependent LTP in the hippocampus: bridging the divide. *Molecular brain* **6**, 1 (2013).

117. Costa, R. P., Padamsey, Z., D'Amour, J. A., Emptage, N. J., Froemke, R. C. & Vogels, T. P. Synaptic transmission optimization predicts expression loci of long-term plasticity. *Neuron* **96**, 177 (2017).

118. Yang, Y. & Calakos, N. Presynaptic long-term plasticity. *Frontiers in Synaptic Neuroscience* **5**, 8 (2013).

119. Castillo, P. E. Presynaptic LTP and LTD of excitatory and inhibitory synapses. *Cold Spring Harbor perspectives in biology* **4**, a005728 (2012).

120. Monday, H. R., Younts, T. J. & Castillo, P. E. Long-term plasticity of neurotransmitter release: emerging mechanisms and contributions to brain function and disease. *Annual review of neuroscience* **41**, 299 (2018).

121. Padamsey, Z. & Emptage, N. Two sides to long-term potentiation: a view towards reconciliation. *Philosophical Transactions of the Royal Society B: Biological Sciences* **369**, 20130154 (2014).

122. Heifets, B. D. & Castillo, P. E. Endocannabinoid signaling and long-term synaptic plasticity. *Annual review of physiology* **71**, 283 (2009).

123. Andrade-Talavera, Y., Duque-Feria, P., Paulsen, O. & Rodrıguez-Moreno, A. Presynaptic spike timing-dependent long-term depression in the mouse hippocampus. *Cerebral Cortex* **26**, 3637 (2016).

124. Cui, Y., Paillé, V., Xu, H., Genet, S., Delord, B., Fino, E., Berry, H. & Venance, L. Endocannabinoids mediate bidirectional striatal spike-timing-dependent plasticity. *The Journal of Physiology* **593**, 2833 (2015).

125. Cui, Y., Prokin, I., Xu, H., Delord, B., Genet, S., Venance, L. & Berry, H. Endocannabinoid dynamics gate spike-timing dependent depression and potentiation. *Elife* **5**, e13185 (2016).

126. Chang, D. T., Honick, A. S. & Reynolds, I. J. Mitochondrial trafficking to synapses in cultured primary cortical neurons. *Journal of Neuroscience* **26**, 7035 (2006).

127. Obashi, K. & Okabe, S. Regulation of mitochondrial dynamics and distribution by synapse position and neuronal activity in the axon. *European Journal of Neuroscience* **38**, 2350 (2013).

128. Sun, T., Qiao, H., Pan, P.-Y., Chen, Y. & Sheng, Z.-H. Motile axonal mitochondria contribute to the variability of presynaptic strength. *Cell reports* **4**, 413 (2013).

129. Lees, R. M., Johnson, J. D. & Ashby, M. C. Presynaptic boutons that contain mitochondria are more stable. *Frontiers in synaptic neuroscience* **11**, 37 (2020).

130. Sjöström, P. J., Turrigiano, G. G. & Nelson, S. B. Rate, timing, and cooperativity jointly determine cortical synaptic plasticity. *Neuron* **32**, 1149 (2001).

131. Sacramento, J., Ponte Costa, R., Bengio, Y. & Senn, W. Dendritic cortical microcircuits approximate the backpropagation algorithm. *Advances in neural information processing systems* **31**, 8721 (2018).

132. Lillicrap, T. P., Cownden, D., Tweed, D. B. & Akerman, C. J. Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications* **7**, 1 (2016).

133. Lee, D.-H., Zhang, S., Fischer, A. & Bengio, Y. *Difference target propagation* in *Joint european conference on machine learning and knowledge discovery in databases* (2015), 498.

134. Olshausen, B. A. & Field, D. J. Sparse coding of sensory inputs. *Current opinion in neurobiology* **14**, 481 (2004).

135. Perez-Orive, J., Mazor, O., Turner, G. C., Cassenaer, S., Wilson, R. I. & Laurent, G. Oscillations and sparsening of odor representations in the mushroom body. *Science* **297**, 359 (2002).

136. Hahnloser, R. H., Kozhevnikov, A. A. & Fee, M. S. An ultra-sparse code underlies the generation of neural sequences in a songbird. *Nature* **419**, 65 (2002).

137. Crochet, S., Poulet, J. F., Kremer, Y. & Petersen, C. C. Synaptic mechanisms underlying sparse coding of active touch. *Neuron* **69**, 1160 (2011).

138. Quiroga, R. Q., Reddy, L., Kreiman, G., Koch, C. & Fried, I. Invariant visual representation by single neurons in the human brain. *Nature* **435**, 1102 (2005).

139. Wixted, J. T., Squire, L. R., Jang, Y., Papesh, M. H., Goldinger, S. D., Kuhn, J. R., Smith, K. A., Treiman, D. M. & Steinmetz, P. N. Sparse and distributed coding of episodic memory in neurons of the human hippocampus. *Proceedings of the National Academy of Sciences* **111**, 9621 (2014).

140. Lodge, M. & Bischofberger, J. Synaptic properties of newly generated granule cells support sparse coding in the adult hippocampus. *Behavioural brain research* **372**, 112036 (2019).

141. McClelland, J. L., McNaughton, B. L. & O'Reilly, R. C. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review* **102**, 419 (1995).

142. Kumaran, D., Hassabis, D. & McClelland, J. L. What learning systems do intelligent agents need? Complementary learning systems theory updated. *Trends in cognitive sciences* **20**, 512 (2016).

143. Buonomano, D. V. & Merzenich, M. M. Cortical plasticity: from synapses to maps. *Annual review of neuroscience* **21**, 149 (1998).

144. Gilbert, C. D., Li, W. & Piech, V. Perceptual learning and adult cortical plasticity. *The Journal of physiology* **587**, 2743 (2009).

145. Moczulska, K. E., Tinter-Thiede, J., Peter, M., Ushakova, L., Wernle, T., Bathellier, B. & Rumpel, S. Dynamics of dendritic spines in the mouse auditory cortex during memory formation and memory recall. *Proceedings of the National Academy of Sciences* **110**, 18315 (2013).

146. Ma, W. J., Beck, J. M., Latham, P. E. & Pouget, A. Bayesian inference with probabilistic population codes. *Nature neuroscience* **9**, 1432 (2006).

147. Fiser, J., Berkes, P., Orbán, G. & Lengyel, M. Statistically optimal perception and learning: from behavior to neural representations. *Trends in cognitive sciences* **14**, 119 (2010).

148. Kappel, D., Habenschuss, S., Legenstein, R. & Maass, W. Network plasticity as Bayesian inference. *PLoS Comput Biol* **11**, e1004485 (2015).

149. Haefner, R. M., Berkes, P. & Fiser, J. Perceptual decision-making as probabilistic inference by neural sampling. *Neuron* **90**, 649 (2016).

150. Wan, L., Zeiler, M., Zhang, S., Le Cun, Y. & Fergus, R. *Regularization of neural networks using dropconnect* in *International conference on machine learning* (2013), 1058.

151. Aitchison, L., Pouget, A. & Latham, P. E. Probabilistic synapses. *arXiv preprint arXiv:1410.1029* (2014).

152. Aitchison, L. & Latham, P. E. Synaptic sampling: A connection between PSP variability and uncertainty explains neurophysiological observations. *arXiv preprint arXiv:1505.04544* (2015).

153. Aitchison, L., Jegminat, J., Menendez, J. A., Pfister, J.-P., Pouget, A. & Latham, P. E. Synaptic plasticity as Bayesian inference. *Nature Neuroscience*, 1 (2021).

154. Fusi, S., Drew, P. J. & Abbott, L. F. Cascade models of synaptically stored memories. *Neuron* **45**, 599 (2005).

155. Roxin, A. & Fusi, S. Efficient partitioning of memory systems and its importance for memory consolidation. *PLoS computational biology* **9**, e1003146 (2013).

156. Benna, M. K. & Fusi, S. Computational principles of synaptic memory consolidation. *Nature neuroscience* **19**, 1697 (2016).

157. Kaplanis, C., Shanahan, M. & Clopath, C. *Continual reinforcement learning with complex synapses* in *International Conference on Machine Learning* (2018), 2497.

158. Isler, K. & van Schaik, C. P. The expensive brain: a framework for explaining evolutionary changes in brain size. *Journal of Human Evolution* **57**, 392 (2009).

159. Navarrete, A., van Schaik, C. P. & Isler, K. Energetics and the evolution of human brain size. *Nature* **480**, 91 (2011).

160. Chen, B. L., Hall, D. H. & Chklovskii, D. B. Wiring optimization can relate neuronal structure and function. *Proceedings of the National Academy of Sciences* **103**, 4723 (2006).

161. Alle, H., Roth, A. & Geiger, J. R. Energy-efficient action potentials in hippocampal mossy fibers. *Science* **325**, 1405 (2009).

162. Sengupta, B., Stemmler, M., Laughlin, S. B. & Niven, J. E. Action potential energy efficiency varies among neuron types in vertebrates and invertebrates. *PLoS Comput Biol* **6**, e1000840 (2010).

163. Perge, J. A., Koch, K., Miller, R., Sterling, P. & Balasubramanian, V. How the optic nerve allocates space, energy capacity, and information. *Journal of Neuroscience* **29**, 7917 (2009).

164. Carter, B. C. & Bean, B. P. Sodium entry during action potentials of mammalian neurons: incomplete inactivation and reduced metabolic efficiency in fast-spiking neurons. *Neuron* **64**, 898 (2009).

165. Hu, H. & Jonas, P. A supercritical density of Na+ channels ensures fast signaling in GABAergic interneuron axons. *Nature neuroscience* **17**, 686 (2014).

166. Martens, J. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193* (2014).

167. Xiao, H., Rasul, K. & Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).

168. Glorot, X. & Bengio, Y. *Understanding the difficulty of training deep feedforward neural networks* in *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (2010), 249.

169. He, K., Zhang, X., Ren, S. & Sun, J. *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification* in *Proceedings of the IEEE international conference on computer vision* (2015), 1026.

170. Grosse, R. & Martens, J. *A kronecker-factored approximate fisher matrix for convolution layers* in *International Conference on Machine Learning* (2016), 573.

171. Ba, J., Grosse, R. & Martens, J. Distributed second-order optimization using Kronecker-factored approximations (2016).

172. Botev, A., Ritter, H. & Barber, D. *Practical gauss-newton optimisation for deep learning* in *International Conference on Machine Learning* (2017), 557.

173. George, T., Laurent, C., Bouthillier, X., Ballas, N. & Vincent, P. Fast approximate natural gradient descent in a kronecker-factored eigenbasis. *arXiv preprint arXiv:1806.03884* (2018).

174. Martens, J., Ba, J. & Johnson, M. *Kronecker-factored curvature approximations for recurrent neural networks* in *International Conference on Learning Representations* (2018).

175. Osawa, K., Tsuji, Y., Ueno, Y., Naruse, A., Yokota, R. & Matsuoka, S. *Large-scale distributed second-order optimization using kronecker-factored approximate curvature for deep convolutional neural networks* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), 12359.

176. Bernacchia, A., Lengyel, M. & Hennequin, G. *Exact natural gradient in deep linear networks and application to the nonlinear case* in (2019).

177. Goldfarb, D., Ren, Y. & Bahamou, A. Practical quasi-newton methods for training deep neural networks. *arXiv preprint arXiv:2006.08877* (2020).

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

179. Ritter, H., Botev, A. & Barber, D. *A scalable laplace approximation for neural networks* in *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings* **6** (2018).

180. Dangel, F., Harmeling, S. & Hennig, P. *Modular block-diagonal curvature approximations for feedforward architectures* in *International Conference on Artificial Intelligence and Statistics* (2020), 799.

181. Zhang, G., Sun, S., Duvenaud, D. & Grosse, R. *Noisy natural gradient as variational inference* in *International Conference on Machine Learning* (2018), 5852.

182. Wu, Y., Mansimov, E., Grosse, R. B., Liao, S. & Ba, J. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. *Advances in neural information processing systems* **30**, 5279 (2017).

183. Grant, E., Finn, C., Levine, S., Darrell, T. & Griffiths, T. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930* (2018).

184. Xiao, H., Rasul, K. & Vollgraf, R. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*

185. Krizhevsky, A. *Learning multiple layers of features from tiny images* tech. rep. (2009).

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

187. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B. & Ng, A. Y. Reading digits in natural images with unsupervised feature learning (2011).

188. Hinton, G. E. & Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *science* **313**, 504 (2006).

189. Zhang, G., Wang, C., Xu, B. & Grosse, R. Three mechanisms of weight decay regularization. *arXiv preprint arXiv:1810.12281* (2018).

190. Amid, E., Anil, R. & Warmuth, M. K. LocoProp: Enhancing BackProp via Local Loss Optimization. *arXiv preprint arXiv:2106.06199* (2021).

191. LeCun, Y. *A theoretical framework for back-propagation* in *Proceedings of the 1988 connectionist models summer school* **1** (1988), 21.

192. Carreira-Perpinan, M. & Wang, W. *Distributed optimization of deeply nested systems* in *Artificial Intelligence and Statistics* (2014), 10.

193. Taylor, G., Burmeister, R., Xu, Z., Singh, B., Patel, A. & Goldstein, T. *Training neural networks without gradients: A scalable admm approach* in *International conference on machine learning* (2016), 2722.

194. Gotmare, A., Thomas, V., Brea, J. & Jaggi, M. Decoupling backpropagation using constrained optimization methods (2018).

195. Ober, S. W. & Aitchison, L. *Global inducing point variational posteriors for bayesian neural networks and deep gaussian processes* in *International Conference on Machine Learning* (2021), 8248.

196. Ollivier, Y. Online natural gradient as a Kalman filter. *Electronic Journal of Statistics* **12**, 2930 (2018).

197. Roux, N., Manzagol, P.-a. & Bengio, Y. Topmoumoute Online Natural Gradient Algorithm. *Advances in Neural Information Processing Systems* **20**, 849 (2007).

198. Ollivier, Y. Riemannian metrics for neural networks I: feedforward networks. *Information and Inference: A Journal of the IMA* **4**, 108 (2015).

199. Ollivier, Y. True asymptotic natural gradient optimization. *arXiv preprint arXiv:1712.08449* (2017).

200. Grosse, R. & Salakhudinov, R. *Scaling up natural gradient by sparsely factorizing the inverse fisher matrix* in *International Conference on Machine Learning* (2015), 2304.

201. Marceau-Caron, G. & Ollivier, Y. Practical Riemannian neural networks. *arXiv preprint arXiv:1602.08007* (2016).

202. Immer, A., Bauer, M., Fortuin, V., Rätsch, G. & Khan, M. E. Scalable marginal likelihood estimation for model selection in deep learning. *arXiv preprint arXiv:2104.04975* (2021).

203. Dangel, F., Tatzel, L. & Hennig, P. ViViT: Curvature access through the generalized Gauss-Newton's low-rank structure. *arXiv preprint arXiv:2106.02624* (2021).

204. Williams, R. J. & Zipser, D. Gradient-based learning algorithms for recurrent. *Backpropagation: Theory, architectures, and applications* **433**, 17 (1995).

205. Tallec, C. & Ollivier, Y. Unbiased online recurrent optimization. *arXiv preprint arXiv:1702.05043* (2017).

206. Mujika, A., Meier, F. & Steger, A. Approximating real-time recurrent learning with random kronecker factors. *arXiv preprint arXiv:1805.10842* (2018).

207. Benzing, F., Gauy, M. M., Mujika, A., Martinsson, A. & Steger, A. *Optimal kronecker-sum approximation of real time recurrent learning* in *International Conference on Machine Learning* (2019), 604.

208. Bengio, Y. How auto-encoders could provide credit assignment in deep networks via target propagation. *arXiv preprint arXiv:1407.7906* (2014).

209. Meulemans, A., Carzaniga, F., Suykens, J., Sacramento, J. & Grewe, B. F. A theoretical framework for target propagation. *Advances in Neural Information Processing Systems* **33**, 20024 (2020).

210. Zhang, G., Martens, J. & Grosse, R. Fast convergence of natural gradient descent for overparameterized neural networks. *arXiv preprint arXiv:1905.10961* (2019).

211. Karakida, R., Akaho, S. & Amari, S.-i. Pathological Spectra of the Fisher Information Metric and Its Variants in Deep Neural Networks. *Neural Computation* **33**, 2274 (2021).

CURRICULUM VITAE

PERSONAL DATA

Name	Frederik Benzing
Date of Birth	February 05, 1994
Place of Birth	Freiburg i. Br., Germany
Citizen of	Germany

EDUCATION

2017 – 2022	ETH Zürich, Zürich, Switzerland *PhD Candidate*
2015 – 2017	ETH Zürich, Zürich, Switzerland *Final degree:* MSc in Mathematics
2012 – 2015	University of Cambridge, Cambridge UK *Final degree:* BA in Mathematics
2010 – 2012	Landesgymnasium für Hochbegabte Schwäbisch Gmünd, Deutschland *Final degree:* Abitur (university entrance diploma)