Rethinking Exponential Averaging of the Fisher

Constantin Octavian Puiu[0000−0002−1724−4533] ✉

University of Oxford, Mathematical Institute,
constantin.puiu@maths.ox.ac.uk

Abstract. In optimization for Machine learning (ML), it is typical that curvature-matrix (CM) estimates rely on an exponential average (EA) of local estimates (giving EA-CM algorithms). This approach has little principled justification, but is very often used in practice. In this paper, we draw a connection between EA-CM algorithms and what we call a *"Wake of Quadratic models"*. The outlined connection allows us to understand what EA-CM algorithms are doing from an optimization perspective. Generalizing from the established connection, we propose a new family of algorithms, *KL-Divergence Wake-Regularized Models* (KLD-WRM). We give three different practical instantiations of KLD-WRM, and show numerically that these outperform K-FAC on MNIST.

Keywords: Optimization · Natural Gradient · KL Divergence · Overfit.

1 Introduction

Recent research in optimization for ML has focused on finding tractable approximations for curvature matrices. In addition to employing ingenious approximating structures ([1,2,3]), curvature matrices are typically estimated as an exponential average (EA) of the previously encountered local estimates ([1,3,4,5,6]). This is in particular true for Natural Gradient (NG) algorithms, which we focus on here. These exponential averages emerge rather heuristically, and have so far only been given incomplete motivations. The main such motivation is *"allowing curvature information to depend on much more data, with exponentially less dependence on older data"* [1]. However, it remains unclear what such an EA algorithm is actually doing from an optimization perspective. In this paper, we show that such EA algorithms can be seen as solving a sequence of local quadratic models whose construction obeys a recursive relationship - which we refer to as a *"Wake of Quadratic Models"*. Inspired by this recursion, we consider a similar, more principled and general recursion for our local models - which we refer to as a *"KL-Divergence Wake-Regularized Models"* (KLD-WRM). We show that under *suitable approximations*, KLD-WRM gives very similar optimization steps to EA-NG. This equivalence raises the hope that using better approximations in our proposed class of algorithms (KLD-WRM) might lead to algorithms which outperform EA-NG. We propose three different practical instantiations of KLD-WRM (of increasing approximation accuracy) and compare them with K-FAC, the most widely used practical implementation of NG for DNNs. Numerically, KLD-WRM outperforms K-FAC on MNIST: with higher test accuracy, lower test loss, and lower variance.

2 Preliminaries

2.1 Neural Networks, Supervised Learning and Notation

We very briefly look at *fully-connected (FC) nets* (we omit CNN for simplicity). We have $\bar{a}_0 := [x, 1]^T$ (x is the input to the net). The *pre-activation* at layer l: $z_l = W_l \bar{a}_{l-1}$ for $l \in \{1, 2, ..., n_L\}$. The *post-activation* at layer l: $a_l = \phi_l(z_l)$ for $l \in \{1, 2, ..., n_L\}$. The *augmented post-activation* at layer l: $\bar{a}_l = [a_l, 1]^T$ for $l \in \{1, 2, ..., n_L - 1\}$ (we augment the post-activations s.t. we can incorporate the bias into the weight matrix W_{l+1} w.l.o.g.; this is standard practice for K-FAC [1]).

In the above, we consider n_L layers, and $\phi_l(\cdot)$ are nonlinear activation functions. We collect all parameters, namely $\{W_l\}_{l=1}^{n_L}$, in a parameter vector $\theta \in \mathbb{R}^d$. Let us consider $N_o \in \mathbb{Z}^+$ neurons in the output layer. The output of the net is $h_\theta(x) := a_{n_L} \in \mathbb{R}^{N_o}$. The predictive (model) distribution of $y|x$ is $p_\theta(y|x) = p_\theta(y|h_\theta(x))$, that is $p(y|x)$ depends on x only through $h_\theta(x)$.

In supervised learning, where we have labeled pairs of datasets $\mathcal{D} = \{(x_i, y_i)\}_i$. Our objective to minimize is typically some regularized modification of

$$f(\theta) = -\log p(\mathcal{D}|\theta) = \sum_{(x_i, y_i) \in \mathcal{D}} \big(-\log p(y_i|h_\theta(x_i)) \big). \tag{1}$$

Thus, we have our loss function $L(y_i, x_i; \theta) := -\log p(y_i|h_\theta(x_i))$, and $f(\theta) = \sum_{(x_i, y_i) \in \mathcal{D}} L(y_i, x_i; \theta)$. This is what we will focus on here, for simplicity of exposition, but our ideas directly apply to different ML paradigms, such as Variational Inference (VI) in Bayesian Neural Networks (BNNs), and to RL as well.

Let us consider the iterates $\{\theta_k\}_{k=0,1,...}$, with θ_0 initialized in standard manner (perhaps on the edge of Chaos)[1]. Let us denote the optimization steps taken at θ_k as s_k, that is $\theta_{k+1} = \theta_k + s_k$. Let $g(\theta)$ and $H(\theta)$ be the gradient and hessian of our objective $f(\theta)$. We will use the notation $g_k := g(\theta_k)$ and $H_k := H(\theta_k)$.

2.2 KL-Divergence and Fisher Information Matrix

The *symmetric*[2] KL-Divergence is a distance measure between two distributions:

$$\mathbb{D}_{KL}(p, q) := \frac{1}{2}\Big[\mathbb{D}_{KL}(q \,||\, p) + \mathbb{D}_{KL}(p \,||\, q)\Big] = \frac{1}{2}\mathbb{E}_{x\sim p}\left[\log \frac{p(x)}{q(x)}\right] + \frac{1}{2}\mathbb{E}_{x\sim q}\left[\log \frac{q(x)}{p(x)}\right]. \tag{2}$$

In our case, we are interested in the symmetric KL-divergence (SKL) between the data joint distribution with one parameter value and the data joint distribution with a different parameter value. We have

$$\mathbb{D}_{KL}(\theta_1, \theta_2) := \mathbb{D}_{KL}\big(p_{\theta_1}(x, y), p_{\theta_2}(x, y)\big). \tag{3}$$

Since we only model the conditional distribution, we let $p_{\theta_i}(x, y) = \hat{q}(x) p(y|h_{\theta_i}(x))$, where $\hat{q}(x)$ is the marginal (empirical) data distribution of x. This gives

[1] Note that the initialization issue is orthogonal to our purpose.

[2] For convenience, we will refer to the *symmetric KL-Divergence* as *SKL-Divergence*.

$$\mathbb{D}_{KL}(\theta_1, \theta_2) = \frac{1}{N} \sum_{x_i \in \mathcal{D}} \mathbb{D}_{KL}\big(p_{\theta_1}(y|x_i), p_{\theta_2}(y|x_i)\big). \tag{4}$$

The Fisher has multiple definitions depending on the situation, but here we have

$$F_k := F(\theta_k) := \mathbb{E}_{\substack{x\sim\hat{q}(x) \\ y\sim p_\theta(y|x)}} \Big[\nabla_\theta \log p_\theta(y|x) \nabla_\theta \log p_\theta(y|x)^T\Big]. \tag{5}$$

This is the Fisher of the joint distribution $p_\theta(x, y) = \hat{q}(x)p(y|h_\theta(x))$. We let $F(\theta_k, x) := \mathbb{E}_{y\sim p_\theta(y|x)}[\nabla_\theta \log p_\theta(y|x) \nabla_\theta \log p_\theta(y|x)^T]$ be the Fisher of the conditional distribution $p_\theta(y|x)$. Then, we have $F_k = \mathbb{E}_{x\sim\hat{q}}[F(\theta_k, x)]$. Using the Fisher, we have the following approximation (see [1,7]) for the SKL-Divergence

$$\mathbb{D}_{KL}\big(p_{\theta_1}(y|x), p_{\theta_2}(y|x)\big) \approx \frac{1}{2}(\theta_2 - \theta_1)^T F(\theta_1, x)(\theta_2 - \theta_1), \tag{6}$$

which is exact as $\theta_2 \to \theta_1$. By plugging (6) into (4) and using linearity we get

$$\mathbb{D}_{KL}(\theta_1, \theta_2) \approx \frac{1}{2}(\theta_2 - \theta_1)^T F_1(\theta_2 - \theta_1). \tag{7}$$

2.3 Natural Gradient and K-FAC

The natural gradient (NG) is defined as [7]

$$\nabla_{NG} f(\theta_k) = F_k^{-1} g_k. \tag{8}$$

The NG descent (NGD) step is then taken to be $s_k^{(NGD)} = -\alpha_k \nabla_{NG} f(\theta_k)$, for some stepsize α_k. NGD has favorable properties, the most notable of which is re-parametrization invariance (when the step-size is infinitesimally small) [8]. The NGD step can be expressed as the solution to the quadratic problem (see [1,8])

$$s_k^{(NGD)} := \arg\min_s s^T g_k + \frac{1}{2\alpha_k} s^T F_k s. \tag{9}$$

K-FAC Storing and inverting the Fisher is prohibitively expensive. K-FAC is a practical implementation of NG which bypasses this problem by approximating the Fisher as a block-diagonal[3] matrix, with each diagonal block further approximated as the Kronecker product of two smaller matrices [1]. That is, we have

$$F_k^{(KFAC)} := \text{blockdiag}\big(\{A_k^{(l)} \otimes \Gamma_k^{(l)}\}_{l=1,\dots,n_L}\big), \tag{10}$$

where each block corresponds to a layer and $\otimes$ denotes the Kronecker product [1]. For example, for FC nets, the Kronecker factors are given by

$$A_k^{(l)} := \mathbb{E}_{x,y\sim p}[\bar{a}_{l-1}\bar{a}_{l-1}^T], \;\; \Gamma_k^{(l)} := \mathbb{E}_{x,y\sim p}[\nabla_{z_l} L \nabla_{z_l} L^T]. \tag{11}$$

Note that the Kronecker factors depend on θ_k which influences both the forward and backward pass. Also, note they can be efficiently worked with since $(A_k^{(l)} \otimes \Gamma_k^{(l)})^{-1} = [A_k^{(l)}]^{-1} \otimes [\Gamma_k^{(l)}]^{-1}$, and $\big([A_k^{(l)}]^{-1} \otimes [\Gamma_k^{(l)}]^{-1}\big)v = \text{vec}\big([\Gamma_k^{(l)}]^{-1} V [A_k^{(l)}]^{-1}\big)$, where v maps to V in the same way $\text{vec}(W_l)$ maps to W_l [1].

[3] Block tri-diagonal approximation is also possible - but this lies outside our scope.

2.4 Curvature in Practice: Exponential Averaging

In practice, many algorithms (including ADAM and K-FAC) do not use the curvature matrix estimate as computed. Instead, they maintain an exponential average (EA) of it (eg. [3,4,5,6]). In the case of NG, this EA is

$$\bar{F}_k := \rho^k F_0 + (1-\rho)\sum_{i=1}^{k} \rho^{k-i} F_i, \tag{12}$$

where $\rho \in [0, 1)$ is the exponential decay parameter. Let us refer to NG algorithms which replace the Fisher, F_k, with its exponential average, $\bar{F}_k$, as EA-NG.

In a similar spirit, K-FAC maintains an EA of the Kronecker factors

$$\bar{A}_k^{(l)} := \rho^k A_0^{(l)} + (1-\rho)\sum_{i=1}^{k} \rho^{k-i} A_i^{(l)}, \quad \bar{\Gamma}_k^{(l)} := \rho^k \Gamma_0^{(l)} + (1-\rho)\sum_{i=1}^{k} \rho^{k-i} \Gamma_i^{(l)}, \tag{13}$$

and in practice, $A_k^{(l)}$ and $\Gamma_k^{(l)}$ in (10) are replaced with $\bar{A}_k^{(l)}$ and $\bar{\Gamma}_k^{(l)}$ respectively. We will refer to the practical implementation of K-FAC which uses EA for the K-FAC matrices as EA-KFAC, to emphasize the presence of the EA aspect. However, this is the norm in practice rather than an exception, and virtually any algorithm referred to as K-FAC (or as using K-FAC) is in fact an EA-KFAC algorithm.

3 A Wake of Quadratic Models (WoQM)

The idea behind WoQM is simple. Instead of taking a step s_k, at θ_k which relies only on the local quadratic model, $s_k = \arg\min_s g^T s + (1/2) s^T B_k s$, we take a step which relies on an EA of all previous local models. Formally, let us define

$$M_k^{(Q)}(s) := g_k^T s + \frac{\lambda_k}{2} s^T B_k s, \tag{14}$$

for an arbitrary *symmetric-positive definite* curvature matrix B_k (typically an approximation of H_k or F_k). Note that $1/\lambda_k$ is a step-size (or learning rate) parameter. Our WoQM step, s_k, is then defined as the solution to

$$\min_s \sum_{i=0}^{k} \rho^{k-i} M_i^{(Q)}\left(s + \sum_{j=i}^{k-1} s_j\right), \text{ with} \tag{15}$$

$$\lambda_0 = \lambda, \text{ and } \lambda_k = (1-\rho)\lambda, \ \forall k \in \mathbb{Z}^+, \tag{16}$$

where we set $\sum_{j=k}^{k-1} s_j = 0$ by convention, $\rho \in [0, 1)$ is an exponential decay parameter, and $\lambda > 0$ is a hyperparameter. While (15) does not appear to be a proper exponential averaging, missing a $1-\rho$ factor in terms where $i \geq 1$, it can be easily rearranged as such by slightly modifying the definition of $M_0^{(Q)}$ (to disobey (14)). To see this, multiply (15) by $(1-\rho)$ and then absorb the $1-\rho$

factor in the definition of $M_0^{(Q)}$. Our stated definition makes the exposition more compact while preserving intuition.

For our choice of model (14), the WOQM step s_k (at θ_k) is the solution of

$$\min_s s^T \left[\sum_{i=0}^{k} \rho^{k-i} \left(g_i + \lambda \kappa(i) B_i \sum_{j=i}^{k-1} s_j \right) \right] + \frac{\lambda}{2} s^T \left[\sum_{i=0}^{k} \kappa(i) \rho^{k-i} B_i \right] s, \tag{17}$$

where we dropped all the constant terms, and have $\kappa(i) := \exp(\mathbb{I}_{\{i>0\}} \log(1-\rho))$, where $\mathbb{I}_{\mathcal{E}}$ is the indicatior function of event $\mathcal{E}$. We use the $\kappa(i)$ term for notational compactness. Note that definition (15) can be used for general models M_i (rather than quadratic $M_i^{(Q)}$), leading to a larger family of algorithms for which WOQM represents a particular instantiation: the *Wake of Models (WoM)* family.

3.1 Connection with Exponential-Averaging in Curvature Matrices

We now look at how the WOQM step relates to Exponential-Averaging Curvature matrices (EA-CM). EA-CM is standard practice in stochastic optimization, and in particular in training DNNs (see for example [1,3,4,5,6]). Formally, using EA-CM boils down to taking a step based on (14), but with B_k replaced by

$$\bar{B}_k := \rho^k B_0 + (1-\rho) \sum_{i=1}^{k} \rho^{k-i} B_i = \sum_{i=0}^{k} \kappa(i) \rho^{k-i} B_i. \tag{18}$$

Note that we used the same exponential decay parameter for convenience, but this is not required.

It is obvious from (17) that we can get an analytic solution for WOQM step s_k, as a function of $\{s_j\}_{j=0}^{k-1}$, $\{(g_j, B_j)\}_{j=0}^{k}$, ρ and λ. Thus, by using the relationship recursively we can get an analytic solution for s_k as a function of $\{(g_j, B_j)\}_{j=0}^{k}$, ρ and λ. When doing this, the connection between WOQM and EA-CM is revealed. The result is presented in *Proposition 3.1.*

Proposition 3.1: Analytic Solution of WOQM step. *The* WOQM *step s_k at iterate θ_k can be expressed as*

$$s_k = -\lambda^{-1} \bar{B}_k^{-1} g_k, \quad \forall k \in \mathbb{Z}^+. \tag{19}$$

Proof. Relies on simple inductive argument. See supplementary material. □

Proposition 3.1 tells us that the WOQM step is exactly the step obtained by using an EA curvature matrix $\bar{B}_k$ in a simple quadratic model of the form (14). That is, WOQM (15)-(16) is the principled optimization formulation of a EA-CM step[4], when the EA-CM stepsize is constant and equal to $1/\lambda$. Thus, we see what EA-CM is actually doing from an optimization perspective: instead of perfectly solving for the local quadratic model, it solves for a trade-off between all previously

[4] WOQM with $\rho = 0$ and B_k based on quantities at θ_k only is also the principled optimization formulation of no-EA CM algorithms, which take steps of the form (14).

encountered models, where the weights of the trade-off are (almost[5]) given by an exponential average (older models receive exponentially less "attention").

There are two observations to make at this point. First, we began by noting that the justification for EA-CM is largely heuristic, but we ended up explaining EA-CM through some optimization model which involved an EA of local models (the WOQM model). Since the EA was the difficult part to justify in the first place, it might seem that we are sweeping the problem from under one rug to another. However, this is not the case. The WOQM formulation aims to reveal a different perspective on EA-CM algorithms, rather than explain the presence of EA itself. It is indeed true that we did not justify why one should use EA and thus get the WOQM family, but this is not required to enhance our understanding and draw conclusions. We can draw conclusions purely based on the established equivalence. This leads us to the second observation, which is why EA-CM improves stability from a stochastic optimization perspective. Rather than conferring stability because it *"uses more data"* ([1,4]), EA-CM can alternatively be thought of as conferring stability because it uses the collection of all previous noisy local models to build a better model[6] (in terms of both noise and functional form).

Note that what we have discussed so far applies when using any curvature matrix B_k. In particular, *Proposition 3.1* can be directly applied to establish an equivalence between EA-NG algorithms and FISHER-WOQM algorithms (WOQM algorithms with $B_k = F_k$). In fact, we can replace the Fisher by any approximation and still have the equivalence holding, *if the EA is done in the form of (12).*

3.2 Fisher-WoQM and Practical K-FAC Equivalence

We have seen (in *Section 2.4*) that K-FAC holds an EA for the *Kronecker factors* (see (13)), rather than for the K-FAC approximation to F_k (as in (12)). Thus, the EA scheme employed by K-FAC is *not* the same as (12) with $F_k \leftarrow F_k^{(KFAC)}$. Therefore, we cannot directly apply *Proposition 3.1* to obtain an equivalence between EA-KFAC and KFAC-WOQM (WOQM with $B_k \leftarrow F_k^{(KFAC)}$). We can loosely establish this equivalence by viewing the *EA over the Kronecker factors* as a convenient (but coarse) approximation to the *EA over* $F_k^{(KFAC)}$ (see *Section 4* in the *supplementary material*). Indeed, carrying an EA for $F_k^{(KFAC)}$ is impractical.

Our equivalence reveals that EA-NG is actually solving an exponentially decaying wake of quadratic models (WOQM) where the curvature matrix (meant to be a Hessian approximation) is taken to be an approximate Fisher. This is in contrast with the typical EA-NG interpretation which says that we take NG steps by solving quadratic models of the form (14) with $B_k = F_k$, but then we further approximate F_k as an EA based on $\{\hat{F}_j\}_{j=1}^{k-1}$. While EA-KFAC does not do the exact same thing, it can be seen as a (very crude) approximation to it.

[5] Can modify the definition of $M_0^{(Q)}$ s.t. WOQM is a proper EA of quadratic models.

[6] Although it is still not clear why putting the previous local models together in an exponentially-averaged fashion is *"right"* for conferring further stability - and this aspect remains heuristic. While this could be informally and partly explained by *"older models should matter less"*, the complete explanation remains an open question.

Dealing with Infrequent Updates In practice, the curvature matrix is not computed at each location, and the EA update is typically performed every $N_u \approx 100$ steps[7] to save computation cost (at least in supervised learning[8]) [1]. In this circumstance, the final implementation of EA-NG would actually be equivalent to a WoQM algorithm where we set

$$B_k = \begin{cases} \hat{F}_k, & \text{if} \quad \text{mod}\ (k, N_u) = 0 \\ \bar{B}_q, \text{ where } q := N_u \lfloor \frac{k}{N_u} \rfloor, & \text{otherwise} \end{cases} . \tag{20}$$

In (20), $\hat{F}_k$ represents an approximation for the Fisher F_k, whose computation has an associated cost. Recall that $\bar{B}_k = \sum_{j=0}^{k} \kappa(j)\rho^{k-j}B_j$. Note that (20) is well defined since $\bar{B}_0 = B_0 = \hat{F}_0$, then $B_1 = B_2 = ... = B_{N_u-1} = B_0$, and hence $\bar{B}_1 = \bar{B}_2 = ... = \bar{B}_{N_u-1} = B_0$, and so on. Further note that by (19), defining our WoQM curvature matirx as in (20) gives us the EA-NG matrix that we want, since we can easily see that: $\bar{B}_{N_u} = \rho\bar{B}_0 + (1-\rho)\hat{F}_{N_u} = \rho\hat{F}_0 + (1-\rho)\hat{F}_{N_u}$. We can then easily extend the argument to show $\bar{B}_{qN_u} = \sum_{j=0}^{q} \kappa(j)\rho^{q-j}\hat{F}_{jN_u}^{(KFAC)}$ which is exactly the form of EA that EA-NG employs when updating statistics every N_u steps. Note that we can apply *Proposition 3.1* irrespectively of our choice of B_k, so in particular, it must hold for B_k as defined in (20).

By extending our reasoning, we see that any heuristic which adapts N_u as a function of observations up until k can be transformed into an equivalent heuristic of picking between $B_k = \bar{B}_q$ (where q is now more generally the previous location where we computed $\hat{F}_k$) and $B_k = \hat{F}_k$. Thus, the equivalence between WoQM and EA-NG holds irrespectively of the heuristic which decides when to update the EA-NG matrix. The exact same reasoning holds for generic EA-CM algorithms.

3.3 Fisher-WoQM: A Different Perspective

Since we have the following approximation[9] for $k \geq i$ [8]

$$\mathbb{D}_{KL}(\theta_i, \theta_k + s) \approx \tilde{\mathbb{D}}_{KL}(\theta_i, \theta_k + s) := \frac{1}{2}\left(s + \sum_{j=i}^{k-1} s_j\right)^T F_i \left(s + \sum_{j=i}^{k-1} s_j\right), \tag{21}$$

one might think that a WoQM model with $B_k = F_k$ and $\rho \in (0, 1)$ would in fact be some form of approximate *"KL-Divergence Wake-Regularized*[10] *model"*. This is indeed true, as we can write our FISHER-WoQM as

$$\min_s \left[\sum_{i=0}^{k} \rho^{k-i} g_i\right]^T s + \lambda \sum_{i=0}^{k} \kappa(i)\rho^{k-i}\tilde{\mathbb{D}}_{KL}(\theta_i, \theta_k + s). \tag{22}$$

Note that $\tilde{\mathbb{D}}_{KL}(\theta_i, \theta_k + s)$ in (21) is a second-order approximation of the

[7] More complicated heuristics can be designed, see [9].

[8] In RL we may prefer updating the K-FAC EA-matrix at each step [10].

[9] Which is exact in the limit as $\theta_k + s \to \theta_i$, but would nevertheless be very crude in practice, particularly for large $k - i$, since the steps taken might be relatively large.

[10] Regularizing w.r.t. SKL divergences relative to all previous distributions, as opposed to just the most recent one - as the quadratic model associated with NG step does.

SKL-Divergence between $p_{\theta_i}(x, y)$ and $p_{\theta_k+s}(x, y)$. Thus, we see that FISHER-WOQM (same as EA-NG by *Prop. 3.1*) is in fact solving a *regularized linear model*, where the model gradient is taken to be the *momentum gradient* (with parameter ρ), and the regularization term is an exponentially decaying wake of *(crudely) approximate* SKL-divergences relative to previously encountered distributions.

This equivalence raises scope for a new family of algorithms: perhaps using another model for the objective f, rather than a simple linear model based on momentum-gradient, and/or using a better approximation for the KL divergence could lead to better performance. This is the main topic of this paper, explored formally in *Section 4* and numerically in *Section 5*. We now note that WOQM (and thus also EA-NG by *Proposition 3.1*, and approximately so, EA-KFAC) is in fact a particular instantiation of the family proposed in the next section.

4 A KL-Divergence Wake-Regularized Models Approach

We now propose a new family of algorithms, which we call *KL-Divergence Wake-Regularized Models (*KLD-WRM*; reads: "Cold-Worm")*. At each location θ_k, the KLD-WRM step s_k is defined as the solution to the problem

$$\min_s M(s; \mathcal{F}_k) + \lambda \sum_{i=0}^{k} \zeta(i)\rho^{k-i}\mathbb{D}_{KL}(\theta_i, \theta_k + s), \tag{23}$$

where $\rho \in [0, 1)$, $M(s; \mathcal{F}_k)$ is a model of the objective which uses *at most* all the information ($\mathcal{F}_k$) encountered up until and including θ_k, and $\zeta(i)$ allows for different 'λ's' at different i's. Simple choices would be $\zeta(i) \equiv 1$, or $\zeta(i) = \kappa(i)$.

The motivation behind KLD-WRM is two-fold. First, a wake of SKL regularization allows us to stay close (in a KL sense) to all previously encountered distributions, rather than only to the most recent one. Thus, we might expect KLD-WRM to give more conservative steps in terms of distribution ($p_\theta(y|x)$) change. Second, KLD-WRM can be seen as a generalization[11] of EA-NG (which is also FISHER-WOQM), which also *"undoes"* the approximation[12] of SKL.

Note that (23) is the most general formulation of KLD-WRM, but in order to obtain practical algorithms we have to make further approximations and definitions (of M and ζ). For example, we could set

$$M(s; \mathcal{F}_k) = \sum_{i=0}^{k} \nu^{k-i} M_i\left(s + \sum_{j=i}^{k-1} s_j\right) \tag{24}$$

where the models $M_i(s)$ are general models (not necessary quadratic), constructed only based on local information at θ_i. This would be a *Wake of Models* KLD-WRM

[11] The linear model $\left[\sum_{i=0}^{k} \rho^{k-i} g_i\right]^T s$ in (22) becomes arbitrary, and $\kappa(i)$ is replaced by a general $\zeta : \mathbb{Z}^+ \to \mathbb{R}$.

[12] For small $\|s\|$ we have $\mathbb{D}(\theta, \theta + s) \approx \mathbb{D}(\theta||\theta + s) \approx \mathbb{D}(\theta + s||\theta) \approx (1/2)s^T F(\theta)s$. Thus, the generalization towards our family could use $\xi\mathbb{D}(\theta||\theta + s) + (1 - \xi)\mathbb{D}(\theta + s||\theta)$ $\forall \xi \in \mathbb{R}$, instead of the SKL (i.e. $\xi = 1/2$). We choose SKL for simplicity.

(WOM-KLD-WRM). We could make this even more particular, for example by considering instantiations where $\nu = 0$, but $\rho \in [0, 1)$ in (24). This would give a *Local-Model* KLD-WRM (LM-KLD-WRM), whose steps s_k are the solution to

$$\min_s M_k(s) + \lambda \sum_{i=0}^{k} \zeta(i)\rho^{k-i}\mathbb{D}_{KL}(\theta_i, \theta_k + s), \tag{25}$$

In this paper, we focus on LM-KLD-WRM instantiations (a particular sub-family of KLD-WRM), and leave the general case as future work. We give three instantiations of LM-KLD-WRM of increasing complexity, and discuss the links with already existing methods. We investigate their performance in *Section 5*.

4.1 Connection between KLD-WRM and Fisher-WoQM

It is easy to see that setting $\nu = \rho$, $\zeta(i) = \kappa(i)$, $M_i(s) = s^T g_i$ and approximating $\mathbb{D}_{KL}(\theta_i, \theta_k + s) \approx \tilde{\mathbb{D}}_{KL}(\theta_i, \theta_k + s)$ (defined in (21)) in a WOM-KLD-WRM gives the WOQM family. Note that WOQM is *not* an LM-KLD-WRM model as $M_k(s)$ can only include information local to θ_k (eg. cannot include g_{k-1}). However, the simplest instantiation of LM-KLD-WRM takes steps which are formally similar to FISHER-WOQM (and thus EA-NG) steps. We investigate this in *Section 4.2*.

4.2 Simplest KLD-WRM Instantiation: Smallest Order KLD-WRM

The simplest practical instantiation of KLD-WRM, *Smallest Order* KLD-WRM (SO-KLD-WRM), uses the most crude approximations to LM-KLD-WRM (25) and sets $\zeta(i) = \kappa(i)$. The SO-KLD-WRM step s_k (at θ_k) is given by

$$\min_s g_k^T s + \lambda \sum_{i=0}^{k} \kappa(i)\rho^{k-i}\tilde{\mathbb{D}}_{KL}(\theta_i, \theta_k + s), \tag{26}$$

It is trivial to see that SO-KLD-WRM differs from FISHER-WOQM (22) only through replacing $\left[\sum_{i=0}^{k} \rho^{k-i} g_i\right]$ with g_k. Since FISHER-WOQM is equivalent to EA-NG, one might expect that SO-KLD-WRM steps are *formally* similar to EA-NG steps. *Proposition 4.1* formalizes this result.

Proposition 4.1: Analytic Solution of SO-KLD-WRM step. *The* SO-KLD-WRM *step s_k at iterate θ_k can be expressed as*

$$s_k = -\lambda^{-1}\bar{F}_k^{-1}[g_k - \rho g_{k-1}], \tag{27}$$

$\forall k \in \mathbb{Z}^+$, where $\bar{F}_k := \sum_{i=0}^{k} \kappa(i)\rho^{k-i}F_i$ and we set $g_{-1} := 0$ by convention.

Proof. By induction. See the *supplementary material*. □

Note that we set $g_{-1} = 0$ to avoid providing two separate cases (for $k = 0$ and for $k \geq 1$). *Proposition 4.1* tells us that the SO-KLD-WRM step is *formally similar* to the FISHER-WOQM step (which we have seen is the EA-NG step). The only (formal) difference between the SO-KLD-WRM step and the EA-NG step is that g_k gets replaced by $g_k - \rho g_{k-1}$ in (19). By *"formally"* here, we mean that the expressions look very similar. However, when considering the two implemented algorithms, the paths taken can be very different.

We have seen that EA-NG (being equivalent to FISHER-WOQM) algorithms are in fact a sub-family of the KLD-WRM family, and obviously SO-KLD-WRM is also a sub-family of KLD-WRM. Thus, the formal difference in the steps between SO-KLD-WRM and EA-NG tells us these two sub-families are distinct. The formal similarity between SO-KLD-WRM and EA-NG, combined with the fact that both are members of the KLD-WRM family raises hopes that more accurate instantiations KLD-WRM might lead to better performance than EA-NG.

Note that basing our SO-KLD-WRM step computation on *Proposition 4.1* gives a tractable algorithm, while solving (26) directly gives an intractable algorithm. To see this, review definition (21), and realize that solving equation (26) *directly* requires storing all previous $\bar{F}_j s_j$ matrix-vector products as a minimum (if the algorithm is written efficiently). Thus, we have an exploding number of vectors that need to be stored when solving (26) *directly*, which eventually will overflow the memory - giving an untractable algorithm. Conversely, using *Proposition 4.1* only requires storing at most 2 matrices ($\bar{F}_k$ and F_k) and 2 gradient-shaped vectors at any one moment in time. Thus, using *Proposition 4.1* with tractable approximations for F_k gives a tractable algorithm.

Reconsidering Gradient Momentum in EA-KFAC By comparing (22) and (26), we see that FISHER-WOQM (which is also EA-NG) can *almost* be seen as SO-KLD-WRM with added momentum for the gradient. The equivalence would be exact if we would have a $\kappa(i)$ inside the sum of the linear term of (22). In EA-KFAC, gradient momentum is not added in the standard fashion. A different way to add momentum is proposed[13] and presented as successful, perhaps because trying to add gradient momentum in the standard fashion gives worse performance [1]. From our discussion, this could be because EA-KFAC can be interpreted as an approximate EA-NG, and EA-NG is a SO-KLD-WRM algorithm which already has included gradient momentum. Thus, further adding momentum does not make sense. Note that by adding momentum to gradient in the *standard fashion* we mean replacing g_k by $\sum_{i=0}^{k} \kappa(i)\rho^{k-i} g_k$ (see [3]).

SO-KLD-WRM in practice When implementing SO-KLD-WRM in practice, we use the K-FAC approximation of the Fisher. Note that *Proposition 4.1* tells us that we need *not* store all previous K-FAC matrices. Instead, we can only save the EA-KFAC matrix. As we have discussed in *Section 3*, we can skip computing the K-FAC matrix at some locations to save on computation. To do that, we just pretend the new K-FAC matrix at θ_k is the EA-KFAC that we currently have stored. For example, if we want to compute K-FAC matrices only once in N_u steps, we use (20), but different heuristics can also be used (for eg. as in [9]). Of course, in practice we store an EA for the Kronecker factors (instead of an EA for the K-FAC matrix), as is standard with practical K-FAC implementation [1].

[13] More akin to a subspace method rather than a standard momentum method (see [1]).

4.3 Second KLD-WRM Instantiation: Q-KLD-WRM

The *Quadratic* KLD-WRM (Q-KLD-WRM) is an LM-KLD-WRM instantiation which uses a second-order approximation for both the Model and the SKLs, and sets $\zeta(i) = \kappa(i)$. The Q-KLD-WRM step s_k (at θ_k) solves

$$\min_s g_k^T s + \frac{1}{2} s^T B_k s + \lambda \sum_{i=0}^{k} \kappa(i) \rho^{k-i} \tilde{\mathbb{D}}_{KL}(\theta_i, \theta_k + s), \tag{28}$$

where B_k is a curvature matrix which aims to approximate the Hessian H_k. That is, Q-KLD-WRM sets M_k in (25) to be a quadratic model. Since (28) is overall a quadratic, we can obtian an *analytic solution* for the Q-KLD-WRM step.

Proposition 4.2: Analytic Solution of Q-KLD-WRM step. *The* Q-KLD-WRM *step s_k at location θ_k is given by the solution to the problem*

$$\min_s s^T \hat{g}_k + \frac{\lambda}{2} s^T \left[\bar{F}_k + \frac{1}{\lambda} B_k \right] s \tag{29}$$

where $\hat{g}_k$ is given by the one-step recursion

$$\hat{g}_{k+1} = g_{k+1} + \rho(I - \hat{M}_k)\hat{g}_k - \rho g_k, \quad \forall k \in \mathbb{Z}^+, \tag{30}$$

with $\hat{g}_0 := g_0$, $\bar{F}_k := \sum_{i=0}^{k} \kappa(i)\rho^{k-i} F_i$, and $\hat{M}_k := \left[I + \frac{1}{\lambda} B_k \bar{F}_k^{-1} \right]^{-1}$. That is, the Q-KLD-WRM *step is formally given by $s_k = -\frac{1}{\lambda}\left[\bar{F}_k + \frac{1}{\lambda} B_k \right]^{-1} \hat{g}_k$.*

Proof. By induction. See the *supplementary material*. □

By comparing *Propositions 4.1* and *4.2*, we see that, unlike SO-KLD-WRM, the Q-KLD-WRM step deviates significantly from the FISHER-WOQM step (also the EA-NG step). Note that setting $B_k = 0$ in *Proposition 4.2* gives *Proposition 4.1*. That is, SO-KLD-WRM is a particular case of Q-KLD-WRM (with $B_k \equiv 0$).

Note that $\{\hat{g}_k\}$ could explode with k, leading to divergence. A sufficient condition for $\{\hat{g}_k\}$ to stay bounded is $\rho \left\| I - \hat{M}_k \right\|_2 \leq \delta$, $\forall k \in \mathbb{Z}^+$, with $\delta \in (0, 1)$ and that $\|\nabla f(\theta)\|_2 \leq K_g$, $\forall \theta$. Under this condition, one can see that $\|\hat{g}_k\|_2 \leq \|g_k - \rho g_{k-1}\|_2 + \delta \|\hat{g}_{k-1}\|_2$. Applying this inequality recursively, and noting that $\|g_k - \rho g_{k-1}\|_2 \leq (1 + \rho) K_g$, we get that our sufficient condition yields $\|\hat{g}_k\|_2 \leq \frac{2}{1-\delta} K_g$, $\forall k \in \mathbb{Z}^+$. The condition $\rho \left\| I - \hat{M}_k \right\|_2 = \rho \left\| I - \left[I + \frac{1}{\lambda} B_k \bar{F}_k^{-1} \right]^{-1} \right\|_2 \leq \delta$ can always be achieved in practice, since taking $\lambda \to \infty$ or $\rho \to 0$ gives $\delta = 0$.

In a similar fashion to the role of *Proposition 4.1*, the role of *Proposition 4.2* (besides highlighting any similarity or dissimilarity to EA-NG) is to give a tractable algorithm. Again, as with SO-KLD-WRM, Q-KLD-WRM is not tractable if we implement it by solving (28) directly for the same reasons. On the other hand, implementing *Proposition 4.2* requires simultaneous storage of *at most* 3 matrices and 3 gradient-shaped vectors at any one point in time - that is, the storage cost does not explode with k. However, because we now have two different sets of matrices involved: $\{F_k\}$ and $\{B_k\}$, the situation is more subtle. In particular, if

B_k and F_k[14] are block-diagonal, then we can see that all involved matrices are block-diagonal, and thus tractably storable and invertable (eg. choose $B_k = F_k$, and approximate $F_k \approx \hat{F}_k^{(KFAC)}$[15]). On the contrary, B_k and F_k might be tractably storable and invertible in isolation, but if they have different structures, computing s_k from *Proposition 4.2* might be intractable.

Q-KLD-WRM in practice In practice, we can in principle employ one of two approximations for B_k. The first option is to use a *BFGS* approximation ([11], [12]). The second option is to replace B_k by the K-FAC matrix. This latter approximation can be justified through the qualified equivalence between the *Fisher* and the *Generalized Gauss-Newton (GGN)* matrix, the latter of which is an approximation to the Hessian. However, in order for the qualified equivalence to hold, we need our predictive distribution $p(y|h_\theta(x))$ to be in the exponential family with natural parameters $h_\theta(x)$ (thinking of each conditional $y|x$ as a different distribution with its own parameters; see [8]). This qualified equivalence holds for most practically used models (see [8]), so is not of huge practical concern.

As we have discussed, it is not obvious how one could get a tractable algorithm from *Proposition 4.2* if the structures of B_k and F_k are dissimilar. Thus, in this paper we focus on instantiations where $B_k := F_k \approx F_k^{(KFAC)}$. In our experiments, we will choose $p(y|h_\theta(x))$ such that the qualified equivalence between Fisher and GGN matrix holds, and thus use $B_k \approx F_k^{(KFAC)}$, $F_k \approx F_k^{(KFAC)}$. As is typical with K-FAC, we also choose to store an EA for the Kronecker factors, rather than for the block-diagonal matrix. With these choices, one can efficiently compute the Q-KLD-WRM step form *Proposition 4.2* (details in the *supplementary material*).

4.4 Third KLD-WRM Instantiation: QE-KLD-WRM

The *Quadratic Objective Approximation Exact SKL*[16] KLD-WRM (QE-KLD-WRM) is the final instantiation of LM-KLD-WRM which we propose here. The QE-KLD-WRM step s_k (at θ_k) solves

$$\min_s g_k^T s + \frac{1}{2} s^T B_k s + \lambda \sum_{i=0}^{k} \zeta(i) \rho^{k-i} \mathbb{D}_{KL}(\theta_i, \theta_k + s), \tag{31}$$

where B_k is a curvature matrix at θ_k, treated the same as we did in Q-KLD-WRM.

Practical QE-KLD-WRM for Regression To be able to work with the exact SKL, we restrict ourselves to a class of $p_\theta(y|x)$ models where the SKL can be expressed in terms of euclidean norms of differences in the network output space. Consider equation (4). For predictive distributions of the form (which are used in regression)

$$p(y|h_{\theta_k}(x_j)) = \mathcal{N}(y|h_{\theta_k}(x_j); I), \tag{32}$$

[14] Of course, $\bar{F}_k$ will have the same structure as F_k.
[15] Using K-FAC further reduces the storage and computation cost through the K-factors.
[16] Note there is no *tilde* on $\mathbb{D}$ in (31

we have a special form for $D_{KL}\big(p(y|h_{\theta_1}(x_j)), p(y|h_{\theta_2}(x_j))\big)$, namely

$$\mathbb{D}_{KL}\big(p(y|h_{\theta_1}(x_j)), p(y|h_{\theta_2}(x_j))\big) = \frac{1}{2}\left\|h_{\theta_1}(x_j) - h_{\theta_2}(x_j)\right\|_2^2. \tag{33}$$

See the *supplementary material* for derivation details. Note that predictive distributions of the form (32) are the most frequently used in practice with regression. For this choice of predictive distribution, $\mathbb{D}_{KL}(\theta_1, \theta_2)$ becomes

$$\mathbb{D}_{KL}(\theta_1, \theta_2) = \frac{1}{2N}\sum_{j=0}^{N}\left\|h_{\theta_1}(x_j) - h_{\theta_2}(x_j)\right\|_2^2. \tag{34}$$

Thus, the QE-KLD-WRM step solves

$$\min_s g_k^T s + \frac{1}{2}s^T B_k s + \frac{\lambda}{2N}\sum_{i=0}^{k}\zeta(i)\rho^{k-i}\sum_{j=0}^{N}\left\|h_{\theta_i}(x_j) - h_{\theta_k+s}(x_j)\right\|_2^2. \tag{35}$$

Practical QE-KLD-WRM for Classification A similar practical computation of $\mathbb{D}_{KL}$ is also available for classification (see the *supplementary material*).

QE-KLD-WRM in Practice While the KL-regularization term in (35) is in principle computable, the amount of associated storage would explode with k (storing old parameters). To bypass this problem, we have two options. We could either choose to discard very old regularization terms, or model them through the approximation (21) (as we did for Q-KLD-WRM, but now only do so for old terms). The latter approach, while convenient, is not well principled - we should really use second-order approximation when we are close (so for recent distributions), not when we are far away. This can be improved, but is left as future work. In this paper, we focus on implementations that use the former approach. Note that we now need to iteratively solve (35) (for eg. with SGD), and get an approximate QE-KLD-WRM step. Further note that the Q-KLD-WRM step given by *Proposition 4.2* is an approximation of the solution to (35), where the SKL-divergence is approximated by its second-order Taylor expansion (21). Thus, the Q-KLD-WRM step is a good initial guess for (35), and we exploit this fact in practice.

Extension to Variable Stepsize For WOQM, SO-KLD-WRM and Q-KLD-WRM, we have so far considered only cases when the *"step-size"* $1/\lambda$ is fixed across different locations θ_k. Indeed, our established equivalence between FISHER-WOQM and EA-NG holds for fixed λ only. However, we may desire variable $\lambda \leftarrow \lambda^{(k)}$ in our KLD-WRM algorithms. To incorporate variable $\lambda^{(k)}$ in Q-KLD-WRM, all one has to do is to merely replace (30) with $\hat{g}_{k+1} = g_{k+1} + (\lambda^{(k+1)}/\lambda^{(k)})\rho[\hat{g}_k - g_k - \hat{M}_k\hat{g}_k]$ and all λ's in *Proposition 4.2* with $\lambda^{(k)}$. The version of *Proposition 4.2* which includes variable $\lambda^{(k)}$ can be found in the *supplementary material*. *Proposition 4.1* with variable $\lambda^{(k)}$ then follows trivially as a particular case with $B_k \equiv 0$.

4.5 Connection with Second Order Methods (SOM)

The particular KLD-WRM instantiation in equation (28), most simply illustrates the connection between our proposed class of algorithms and SOM. Setting $\lambda \leftarrow 0$ in (28) reverses Q-KLD-WRM back to a simple second order algorithm. More generally, from (23) we see that relative to SOM, KLD-WRM generalizes the local second order model to an arbitrary model that may also include previous information (in principle), and more importantly, it adds a wake of $\mathbb{D}_{KL}$ regularization. The connection between NG and Generalized Gauss-Newton can be found in [8].

5 Numerical Results

We compare our proposed KLD-WRM instantiations (SO, Q and QE) with K-FAC, on the MNIST classification problem. We investigated 4 different hyper-parameter settings for each solver, but only present the best ones here. The complete results, implementation details, as well as more in depth discussion can be found in *Section 8* of the *supplementary material*. *Figures 1 and 2* show *test loss* and *test accuracy* for our considered solvers. Ten runs were performed for each solver. Important summary statistics of these results are shown in *Table 1*.

Hyper-parameters: The values of ρ and λ are specified in *Figures 1* and *2*. We used $\zeta(i) = \kappa(i)/330$ for QE, and $\zeta(i) = \kappa(i)$ for the other solvers, because the QE estimates of the SKL term were larger. The QE-specific hyper-parameters are: ω/λ – the learning rate of the inner SGD solving (35), N_{IS} – the number of inner SGD steps per iteration, and N_{CAP} – the total number of networks stored. For the results presented here, these were set to $7 \cdot 10^{-4}$, 10 and 4 respectively.

Test Accuracy and Loss: All KLD-WRM variants exceed 97.5% mean test accuracy, outperforming K-FAC by about 1.5%. The SD is 4-5 times lower for KLD-WRM variants, which is desirable. Analogous observations hold for the test loss. Since higher variance may more frequently yield *"favorable"* outliers, we show relevant metrics for this aspect in *Columns 2-6* of *Table 1*. We see that even from the *favorable outliers* point of view, KLD-WRM is mostly preferable.

Robustness and Overall Performance: *If we can only run the training a few times (perhaps once)*, it is preferable (in terms of epochs; both from a *test accuracy* and *test-loss* point of view) to use KLD-WRM rather than K-FAC.

Table 1. MNIST results summary. SO, Q and QE refer to KLD-WRM variants. *"Accuracy"* and *"loss"* are the test-set ones. μ_{acc} and σ_{acc} are the mean and SD of the empirical distribution of *accuracy* at the end of epoch 50. Notation is analogous for μ_{loss} and σ_{loss}, which refer to the *loss*. $\mathcal{N}_{\mathcal{C}}$ is the number of runs for which condition $\mathcal{C}$ is satisfied.

	$\mathcal{N}_{acc \geq 98\%}$	$\mathcal{N}_{acc > 98\%}$	$\mathcal{N}_{acc \geq 98.5\%}$	$\mathcal{N}_{loss \leq 0.25}$	$\mathcal{N}_{loss \leq 0.2}$	μ_{acc}	σ_{acc}	μ_{loss}	σ_{loss}
K-FAC	3	3	**1**	5	**2**	96.19%	3.2%	0.26	0.10
SO	4	2	0	7	0	97.60%	0.85%	0.25	0.04
Q	5	**4**	0	4	0	97.69%	0.69%	0.26	0.04
QE	**8**	**7**	**2**	**9**	0	**98.01%**	**0.64%**	**0.23**	**0.02**

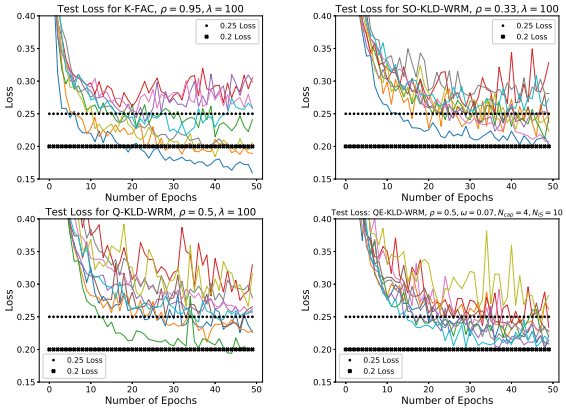

Fig. 1. MNIST test-loss results for K-FAC, and our three KLD-WRM variants.

That is because all our KLD-WRM variants more robustly achieve good test metrics. Conversely, *if we can run the training many times* (and choose the best run) K-FAC's large variance *may* eventually play to our advantage (not guaranteed).

KLD-WRM variant selection and winners: SO-KLD-WRM and Q-KLD-WRM have virtually the same computation cost per epoch as K-FAC. Conversely, QE-KLD-WRM has the same *data acquisition* and *linear algebra* costs as K-FAC, but 3-10 times higher oracle cost (fwd. and bwd. pass cost), owing to approximately solving (35). Thus, when *data cost is relatively low*, SO-KLD-WRM and Q-KLD-WRM will be preferable, as they will have the smallest wall-time per epoch (while having almost the same performance per epoch as QE-KLD-WRM). Conversely, when *data cost dominates*, all 4 solvers will have the same wall-time per epoch. In this case, QE-KLD-WRM is preferable as it gives the best performance per epoch[17].

6 Conclusions and Future Work

We established an equivalence between EA-CM algorithms (typically used in ML) and WOQM algorithms (which we defined in *Section 3*). The equivalence revealed what EA-CM algorithms are doing from a model-based optimization point of view. Generalizing from WOQM, we defined a broader class of algorithms in *Section 4*: KLD-WRM. We then focused our attention on a different subclass of KLD-WRM, LM-KLD-WRM, and provided three practical instantiations of it. Numerical results on MNIST showed that performance-metrics distributions have better mean and lower variance for our KLD-WRM algorithms, indicating they are preferable to K-FAC in practice due to higher robustness.

Future work: (a) KLD-WRM for VI BNNs and RL; (b) convergence theory; (c) investigate Q and QE variants when B_k and F_k have different structures; (d) consider KLD-WRM algorithms outside the LM-KLD-WRM subfamily (include info. at $\{\theta_j\}_{j<k}$ in $M(s; \mathcal{F}_k)$; see (23)); (e) consider arbitrary $\xi \in \mathbb{R}$ (see footnote 12).

[17] Codes available at: https://github.com/ConstantinPuiu/Rethinking-EA-of-the-Fisher

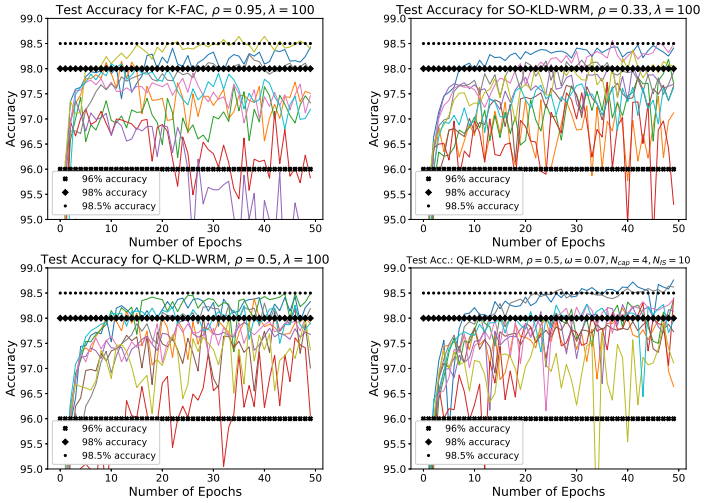

Fig. 2. MNIST test-accuracy results for K-FAC, and our three KLD-WRM variants.

Acknowledgments Thanks to *Jaroslav Fowkes* for very useful discussions. I am funded by the EPSRC CDT in InFoMM (EP/L015803/1) in collaboration with Numerical Algorithms Group and St. Anne's College (Oxford).

References

1. Martens, J.; Grosse, R. Optimizing neural networks with Kronecker-factored approximate curvature, In: arXiv:1503.05671 (2015).
2. Yang, M.; Xu, D; Wen, Z.; Chen, M.; Xu, P. Sketchy empirical natural gradient methods for deep learning, In: arXiv:2006.05924 (2021).
3. Ba, J.; Kingma, D. Adam: A method for stochastic optimization, ICLR (2015).
4. LeCun, Y.; Bottou, L.; Orr, G.; Muller, K. Efficient backprop. Neural networks: Tricks of the trade, pages 546-546 (1998).
5. Schaul, T.; Zhang, S.; LeCun, Y. No more pesky learning rates. In ICML (2013).
6. Park, H.; Amari, S.-I.; Fukumizu, K. Adaptive natural gradient learning algorithms for various stochastic models. Neural Networks, 13(7):755-764 (2000).
7. Amari, S. I. Natural gradient works efficiently in learning, Neural Computation, 10(20), pp. 251-276 (1998).
8. Martens, J. New insights and perspectives on the natural gradient method, arXiv:1412.1193 (2020).
9. Osawa, K.; Yuichiro Ueno, T.; Naruse, A.; Foo, C.-S.; Yokota, R. Scalable and practical natural gradient for large-scale deep learning, arXiv:2002.06015 (2020).
10. Wu, Y.; Mansimov, E.; Grosse, R. B.; Liao, S.; Ba, J. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In Advances in neural information processing systems, pages 5285-5294 (2017).
11. Bottou, L.; Curtis, F. E.; Nocedal, J. Optimization methods for large-scale machine learning (2018).
12. Goldfarb, D.; Ren, Y.; Bahamou, A. Practical Quasi-Newton methods for training deep neural networks, arXiv:2006.08877, (2021).