FAdam: Adam is a natural gradient optimizer using diagonal empirical Fisher information

Dongseong Hwang
Google LLC
Mountain View, CA, USA
dongseong@google.com

Abstract

This paper establishes a mathematical foundation for the Adam optimizer, elucidating its connection to natural gradient descent through Riemannian and information geometry. We rigorously analyze the diagonal empirical Fisher information matrix (FIM) in Adam, clarifying all detailed approximations and advocating for the use of log probability functions as loss, which should be based on discrete distributions, due to the limitations of empirical FIM. Our analysis uncovers flaws in the original Adam algorithm, leading to proposed corrections such as enhanced momentum calculations, adjusted bias corrections, adaptive epsilon, and gradient clipping. We refine the weight decay term based on our theoretical framework. Our modified algorithm, **Fisher Adam (FAdam)**, demonstrates superior performance across diverse domains including LLM, ASR, and VQ-VAE, achieving state-of-the-art results in ASR.

1 Introduction

Natural Gradient Descent (NGD) is a powerful optimization method that has shown promise in training machine learning models. Introduced by Amari [3], and revisited recently [11], NGD offers an alternative to traditional gradient descent by taking into account the curvature of the loss landscape. This is achieved through the use of the Fisher information matrix (FIM), which provides a Riemannian metric for the statistical manifold. Despite its potential benefits, NGD faces a significant computational challenge: the calculation of FIM, which can be prohibitively expensive for large models like deep neural networks.

Adam [12] is the de facto standard optimizer, favored for its fast convergence and practicality. While the original paper mentions the second momentum term can be interpreted as the diagonal FIM, a comprehensive theoretical understanding of the square root FIM, akin to the approach first seen in AdaGrad [25], has remained elusive [60]. This is in contrast to NGD, which utilizes the inverse FIM.

Our main contribution is to provide an accessible mathematical explanation of the Adam optimizer, drawing upon fundamental concepts from Riemannian geometry and information geometry, thereby demonstrating that Adam is an approximation of NGD. This framework enables us to illuminate the origin of the square root term in Adam and to advocate for the use of a log probability function as the loss function when employing Adam.

We demonstrate that Adam utilizes the diagonal empirical Fisher information, providing a detailed explanation of both the diagonal approximation and the empirical approximation. Notably, the empirical approximation suggests that the loss function should be based on discrete distributions, which implies using categorical cross-entropy loss rather than L2 loss in the image domain. Furthermore, our analysis uncovers flaws in the original Adam, leading us to propose corrections such as averaging natural gradient by momentum, adjusting bias corrections, and introducing adaptive epsilon and

gradient clip. We also enhance the weight decay using principles of Riemannian geometry. Our modified algorithm, named **Fisher Adam (FAdam)**, demonstrates strong performance across various domains, as evidenced by our experiments with Large Language Models (LLMs) for text, Automatic Speech Recognition (ASR) for speech, and Vector Quantized Variational Autoencoder (VQ-VAE) for image. In particular, our ASR experiments achieve new state-of-the-art results.

2 Background

We use the notation from differential geometry, as introduced in Appendix A.1.

2.1 Gradient on Riemannian manifold

The inner product in the tangent space of a Riemannian manifold is defined by the Riemannian metric tensor $\mathrm{g} = g_{ij} d^i \otimes d^j$. The Riemannian metric tensor components, g_{ij}, are symmetric and positive definite. Therefore, g_{ij} is always invertible, and its inverse is denoted as g^{ij}.

$$\vec{v} \cdot \vec{u} := \mathrm{g}(\vec{v}, \vec{u}) = g_{ij} d^i \otimes d^j (v^k \partial_k, u^r \partial_r) = g_{ij} v^k u^r d^i(\partial_k) \otimes d^j(\partial_r) = g_{ij} v^i u^j \tag{1}$$

The directional derivative of a scalar field ϕ along a vector $\vec{v}$ is defined as the dot product of the gradient of ϕ. The directional derivative can also be expressed using the covariant derivative. By rearranging the terms with respect to $\nabla \phi$, we obtain the following equation for the gradient $\nabla \phi$ in the tangent space $\mathcal{T}_{\boldsymbol{\theta}} \mathcal{M}$.

$$\nabla \phi \cdot \vec{v} = (\nabla \phi)^i v^j g_{ij} = D_{\vec{v}} \phi = v^j \frac{\partial \phi}{\partial \theta^j} \tag{2}$$

$$(\nabla \phi)^i = g^{ij} \frac{\partial \phi}{\partial \theta^j} \rightarrow \nabla \phi = g^{ij} \frac{\partial \phi}{\partial \theta^j} \partial_i \tag{3}$$

In Euclidean space, the Riemannian metric g_{ij} is Kronecker delta, so the gradient is usually expressed as follows:

$$\nabla \phi = \frac{\partial \phi}{\partial \theta^i} \partial_i \tag{4}$$

In ML community, equation Eq. (4) is often denoted as the **gradient** $\nabla \phi$, while equation Eq. (3) is denoted as the **natural gradient** $\tilde{\nabla} \phi$. This notation conflicts with conventions in differential geometry. Since our paper is focused on ML, we will adhere to ML conventions from this point forward.

Typically, the component part of tensor operations are represented using matrix multiplication. The metric tensor component, g_{ij}, is represented by a matrix $\boldsymbol{G}$, and $\nabla \phi$ is a column vector.

$$\tilde{\nabla} \phi = \boldsymbol{G}^{-1} \nabla \phi \tag{5}$$

Equation 1 can also be expressed using matrix multiplication.

$$\vec{v} \cdot \vec{u} = \boldsymbol{v}^\top \boldsymbol{G} \boldsymbol{u} \tag{6}$$

2.2 Fisher Information Matrix (FIM) and Natural Gradient

The Fisher information quantifies the amount of information an observable random variable x (representing data) conveys about an unknown parameter $\boldsymbol{\theta}$ (representing a model parameter) influencing its probability. Given the probability mass function $P(\mathrm{x}|\boldsymbol{\theta})$ (or probability density function $p(\mathrm{x}|\boldsymbol{\theta})$) for the random variable x, the Fisher information is defined as the variance of the score function (i.e., the gradient of the log-likelihood), which is symmetric and positive semi-definite by definition.

$$\boldsymbol{F}(\boldsymbol{\theta}) := \mathbb{E}_{\mathrm{x}\sim P_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}} \log P(\mathrm{x}|\boldsymbol{\theta}) \, \nabla_{\boldsymbol{\theta}} \log P(\mathrm{x}|\boldsymbol{\theta})^{\top}]. \tag{7}$$

Fisher information is the expected value of the Hessian matrix. (Proof provided in Appendix A.2.) It represents the curvature of the log-likelihood on the statistical manifold where the model parameters $\boldsymbol{\theta}$ reside.

$$\boldsymbol{F}(\boldsymbol{\theta}) = -\mathbb{E}_{\mathrm{x}\sim P_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}}^2 \log P(\mathrm{x}|\boldsymbol{\theta})] = -\mathbb{E}_{\mathrm{x}\sim P_{\boldsymbol{\theta}}}[\boldsymbol{H}_{\boldsymbol{\theta}}(\log P(\mathrm{x}|\boldsymbol{\theta}))]. \tag{8}$$

In the realm of statistical manifolds, the distance between $\boldsymbol{\theta}$ and $\boldsymbol{\theta} + \boldsymbol{d}$ is quantified by the Kullback-Leibler divergence $D_{\mathrm{KL}}(P(\mathrm{x}|\boldsymbol{\theta})\|P(\mathrm{x}|\boldsymbol{\theta} + \boldsymbol{d}))$. For an infinitesimal displacement $\boldsymbol{d}$, a second-order Taylor series approximation reveals the Fisher information as the underlying distance metric [3, 11, 66]. A detailed proof can be found in Appendix A.3.

$$D_{\mathrm{KL}}(P(\mathrm{x}|\boldsymbol{\theta})\|P(\mathrm{x}|\boldsymbol{\theta} + \boldsymbol{d})) \approx \frac{1}{2}\boldsymbol{d}^{\top}\boldsymbol{F}(\boldsymbol{\theta})\boldsymbol{d}. \tag{9}$$

As mentioned in Eq. (6), the magnitude of a vector $\boldsymbol{v}$ in the tangent space $\mathcal{T}_{\boldsymbol{\theta}}\mathcal{M}$ can be expressed using the inner product, and the Fisher Information Matrix (FIM) serves as the Riemannian metric [63].[1] Prior works [50, 61, 64] have demonstrated the importance of FIM in providing an inherent scale for learning.

$$\vec{v} \cdot \vec{v} = \|\boldsymbol{v}\|^2 = \boldsymbol{v}^{\top}\boldsymbol{G}\boldsymbol{v} = \boldsymbol{v}^{\top}\boldsymbol{F}(\boldsymbol{\theta})\boldsymbol{v} \tag{10}$$

In statistical manifolds, the gradient is described using the Fisher information $\boldsymbol{F}$ in Eq. (5), and this is called the **natural gradient**.

$$\tilde{\nabla}\phi = \boldsymbol{F}^{-1}\nabla\phi \tag{11}$$

The intuitive interpretation of Eq. (11) is that components with higher information undergo conservative movement, while components with lower information exhibit wider movement.

The inverse FIM exhibits a close relationship with the covariance matrix of the log likelihood, denoted as $\Sigma_{\boldsymbol{\theta}}^{-1}$. The Mahalanobis distance, $d_M(\boldsymbol{x}, \mathcal{P})^2$, which measures the distance between a data point $\boldsymbol{x}$ and a distribution $\mathcal{P}$, is defined as $d_M(\boldsymbol{x}, \mathcal{P})^2 := (\boldsymbol{x} - \boldsymbol{\mu})^{\top}\Sigma_{\boldsymbol{x}}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})$ [22]. Comparing this to Eq. (10), we discern a connection between the inverse Fisher information and the covariance matrix: higher covariance indicates lower information. This parallel echoes the principle in information theory where higher entropy corresponds to lower information.

3 Toward FAdam

3.1 Loss and FIM

To utilize the natural gradient Eq. (11), we need to know both the gradient term and FIM. The gradient term requires the calculation of the loss $\mathcal{L}(\boldsymbol{\theta})$, while the FIM requires the score function $\nabla_{\boldsymbol{\theta}} \log P(\mathrm{x}|\boldsymbol{\theta})$. If the loss is expressed in the form of $-\log P(\mathrm{x}|\boldsymbol{\theta})$, we eliminate the need to calculate the score function separately. Therefore, for using natural gradient optimizers like Adam[2] [12], **the loss function must be in the form of the log-likelihood**. We will delve deeper into the choice of loss function in Section 3.3.1.

$$\tilde{\nabla}\mathcal{L}(\boldsymbol{\theta}) = \boldsymbol{F}^{-1}\nabla\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\mathrm{x}\sim P_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}} \log P(\mathrm{x}|\boldsymbol{\theta}) \, \nabla_{\boldsymbol{\theta}} \log P(\mathrm{x}|\boldsymbol{\theta})^{\top}]^{-1}\nabla_{\boldsymbol{\theta}} - \log P(\mathrm{x}|\boldsymbol{\theta}) \tag{12}$$

The model parameter $\boldsymbol{\theta}$ is updated using the given natural gradient $\tilde{\nabla}\mathcal{L}(\boldsymbol{\theta})$, where η is the learning rate.

---

[1]Since FIM is symmetric and positive semi-definite, it produces a Pseudo-Riemannian manifold [63].

[2]We will discuss why Adam is considered a natural gradient optimizer in Section 3.4.

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \boldsymbol{F}^{-1} \nabla \mathcal{L}(\boldsymbol{\theta}) \tag{13}$$

Natural gradient is considered a second-order method because FIM is the expected value of the Hessian, as shown in Eq. (8). A comparison with Newton's method is provided in section Appendix B.1.

3.2 Diagonal Fisher Information

One key reason for Adam [12]'s significant success compared to other second-order methods is its memory complexity. Adam scales linearly with the number of parameters, $O(N)$, while methods using FIM typically scale quadratically, $O(N^2)$. For models with billions of parameters, $O(N^2)$ is impractical.

Adam utilizes the diagonal Fisher information matrix. As discussed in 2.2, the inverse FIM approximates the covariance of the log likelihood. The diagonal FIM results in the loss of all covariance information except for the variances. Interestingly, Adam's success story suggests that the loss of covariance information might not be detrimental in practice.

Let $\hat{\boldsymbol{F}}(\boldsymbol{\theta})$ denote the diagonal FIM obtained by diagonalizing FIM in Eq. (7). The loss function (12) is greatly simplified.

$$\hat{\boldsymbol{F}}(\boldsymbol{\theta}) := \mathbb{E}_{\mathrm{x} \sim P_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}} \log P(\mathrm{x}|\boldsymbol{\theta})^2] \tag{14}$$

$$\tilde{\nabla}\mathcal{L}(\boldsymbol{\theta}) = \hat{\boldsymbol{F}}^{-1}\nabla\mathcal{L}(\boldsymbol{\theta}) = -\frac{\nabla_{\boldsymbol{\theta}}\log P(\mathrm{x}|\boldsymbol{\theta})}{\mathbb{E}_{\mathrm{x} \sim P_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}} \log P(\mathrm{x}|\boldsymbol{\theta})^2]} \tag{15}$$

Amari u.a. [27] proves that the off-diagonal elements of FIM are smaller than the diagonal elements by an order of $1/\sqrt{n}$, where n represents the number of elements in the matrix. This finding justifies the use of the quasi-diagonal natural gradient method when the weight matrices of each layer are sufficiently large like LLM (Large Language Models).

Meanwhile, there have been efforts to capture important off-diagonal elements [29, 58]. K-FAC [15] employs a Kronecker-factored approximation to the FIM. Shampoo [24] utilizes a low-rank approximation. The application of off-diagonal FIM to Adam is left for future study.

3.3 Empirical Fisher Information

To compute the diagonal FIM in Eq. (14), the expected value needs to be calculated with respect to the parametric probabilistic model $P(\mathrm{x}|\boldsymbol{\theta})$. While data can be sampled from the parametric model, it is not always a straightforward process. Various sampling methods exist, such as Gibbs sampling, Langevin Markov chain Monte Carlo (MCMC) sampling [1], Metropolis-Hastings MC sampling [6] and the recent GFlowNet [43]. However, none of these methods are universally efficient in generating sufficient data with high fidelity and diversity.

Instead of $P(\mathrm{x}|\boldsymbol{\theta})$, we utilize the true data-generating distribution $p_{data}(\mathrm{x})$ to compute FIM. Although the exact form of $p_{data}(\mathrm{x})$ is unknown, we have access to a training set of samples. We compute FIM by utilizing the training set $\mathcal{D}$ by substituting the true distribution $p_{data}(\mathrm{x})$ with the empirical distribution $\hat{p}_{data}(\mathrm{x})$. This approximated FIM is referred to as the **empirical FIM** in the statistics community [28]. The training set might not contain enough samples of low-probability x, which could lead to issues with the empirical FIM.

$$\hat{\boldsymbol{F}}(\boldsymbol{\theta}) = \mathbb{E}_{\mathrm{x} \sim P_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}} \log P(\mathrm{x}|\boldsymbol{\theta})^2] \approx \mathbb{E}_{\mathrm{x} \sim p_{data}}[\nabla_{\boldsymbol{\theta}} \log P(\mathrm{x}|\boldsymbol{\theta})^2], \tag{16}$$

$$\approx \mathbb{E}_{\mathrm{x} \sim \hat{p}_{data}}[\nabla_{\boldsymbol{\theta}} \log P(\mathrm{x}|\boldsymbol{\theta})^2] = \frac{1}{|\mathcal{D}|} \sum_{\mathrm{x} \in \mathcal{D}} \nabla_{\boldsymbol{\theta}} \log P(\mathrm{x}|\boldsymbol{\theta})^2 \tag{17}$$

The expected value of the loss function is also obtained from the empirical distribution rather than the true distribution. Optimizing this cost function $J(\boldsymbol{\theta})$ is referred to as empirical risk minimization, for similar reasons.

$$J(\boldsymbol{\theta}) = -\mathbb{E}_{\mathrm{x} \sim p_{data}}[\log P(\mathrm{x}|\boldsymbol{\theta})] \approx -\mathbb{E}_{\mathrm{x} \sim \hat{p}_{data}}[\log P(\mathrm{x}|\boldsymbol{\theta})] = -\frac{1}{|\mathcal{D}|}\sum_{\mathrm{x} \in \mathcal{D}}\log P(\mathrm{x}|\boldsymbol{\theta}) \qquad (18)$$

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}}\mathbb{E}_{\mathrm{x} \in \mathcal{D}}[\mathcal{L}(\boldsymbol{\theta})] = \frac{1}{|\mathcal{D}|}\sum_{\mathrm{x} \in \mathcal{D}}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{|\mathcal{D}|}\sum_{\mathrm{x} \in \mathcal{D}}\nabla_{\boldsymbol{\theta}}\log P(\mathrm{x}|\boldsymbol{\theta}) \qquad (19)$$

Calculating the exact cost function and FIM over the entire training set is computationally expensive. Therefore, in practice, the expected value is approximated using a minibatch $\mathcal{B}$. As this approximation further increases the uncertainty in the empirical FIM, FIM is typically estimated using an exponential moving average (EMA). Therefore, during training, natural gradient is computed as follows:

$$\tilde{\nabla} J(\boldsymbol{\theta}) = \hat{\boldsymbol{F}}^{-1}\nabla J(\boldsymbol{\theta}) \approx -\mathbb{E}_{\mathrm{x} \in \mathcal{B}}[\nabla_{\boldsymbol{\theta}}\log P(\mathrm{x}|\boldsymbol{\theta})]/\mathbb{E}_{EMA}[\mathbb{E}_{\mathrm{x} \in \mathcal{B}}[\nabla_{\boldsymbol{\theta}}\log P(\mathrm{x}|\boldsymbol{\theta})^2]] \qquad (20)$$

$$\approx -\mathbb{E}_{\mathrm{x} \in \mathcal{B}}[\nabla_{\boldsymbol{\theta}}\log P(\mathrm{x}|\boldsymbol{\theta})]/\mathbb{E}_{EMA}[\mathbb{E}_{\mathrm{x} \in \mathcal{B}}[\nabla_{\boldsymbol{\theta}}\log P(\mathrm{x}|\boldsymbol{\theta})]^2] \qquad (21)$$

$$= -\boldsymbol{g}/\mathbb{E}_{EMA}[\boldsymbol{g}^2] \qquad (22)$$

Let the gradient of a minibatch be denoted as $\boldsymbol{g}$. To reuse $\boldsymbol{g}$ for calculating FIM, Adam makes another approximation from Eq. (20) to Eq. (21). Wang u.a. [59] showed that using Eq. (20) makes Adam less sensitive to batch size. Adam variants are recommended to use large batch sizes [17, 53] to accurately estimate not only the gradient but also FIM.

In supervised learning, the loss function becomes conditional log-likelihood, necessitating specific considerations detailed in Appendix B.2.

3.3.1 Discrete vs Continuous probability distributions

It is worth noting that while Adam has demonstrated superior performance to SGD in the text domain, it has been repeatedly reported that its convergence point in the image domain is worse than SGD [26, 33, 41, 42, 53]. Empirical evidence indicates that Adam excels when dealing with discrete distributions, such as text inputs with categorical distributions. However, it may encounter difficulties when handling continuous distributions, such as image inputs with Gaussian distributions.

In the image domain, using the L2 loss is common practice due to its equivalence to the negative log-likelihood of a Gaussian distribution.

$$J(\boldsymbol{\theta}) \approx -\mathbb{E}_{\mathrm{x} \sim \hat{p}_{data}}[\log p(\mathrm{x}|\boldsymbol{\theta})] = -\mathbb{E}_{\mathrm{x} \sim \hat{p}_{data}}\left[k\log\exp\left(\frac{\mathrm{x}-\boldsymbol{\mu}(\boldsymbol{\theta})}{\sigma}\right)^2\right] \qquad (23)$$

$$= -k\mathbb{E}_{\mathrm{x} \sim \hat{p}_{data}}\left[(\mathrm{x}-\boldsymbol{\mu}(\boldsymbol{\theta}))^2\right] = -\frac{k}{|\mathcal{D}|}\sum_{\mathrm{x} \in \mathcal{D}}(\mathrm{x}-\boldsymbol{\mu}(\boldsymbol{\theta}))^2 \qquad (24)$$

We performed empirical approximations to calculate the expected value of FIM. Eq. (16) represents the empirical approximation of FIM for generative models, while Eq. (51) and Eq. (57) represent the empirical approximations for discriminative models. We hypothesize that these empirical approximations cause significantly more problems in continuous distributions than in discrete distributions. This disparity arises from the fundamental difference in how expected values are calculated for discrete and continuous distributions. Discrete distributions rely on probability mass functions, where the softmax function often concentrates the majority of probability mass on a few top logits. This concentration allows for relatively accurate empirical approximations. In contrast, continuous distributions require integration over their probability density functions, making the estimation of their expected values with a single sampled value an overly simplified and potentially inaccurate approximation.

Kunstner u.a. [28] argue against the use of the empirical Fisher in natural gradient descent, providing examples solely with continuous inputs and distributions.[3] Their findings support our assertion that

---

[3]Regrettably, relying on image classification (i.e., continuous inputs) or, even more restrictively, simple regression like curve fitting (i.e., continuous distribution) to analyze Adam or FIM may limit the generalizability of findings [28, 30, 48].

Adam may not perform well with continuous distributions, while the empirical success of Adam suggests that the empirical Fisher is adequate for discrete distributions.

We propose utilizing **log-likelihood loss based on discrete probability distributions**. In the image domain, this translates to using **cross-entropy (CE) loss on a categorical distribution instead of the L2 loss**.[4] This can be implemented by modifying predictions to utilize a one-hot encoding, predicting 256 values per RGB channel. As demonstrated in Section 4.3, this modification significantly enhances the FID (Fréchet Inception Distance) metric for VQ-VAE [18].

It has also been reported that categorical CE loss improves accuracy in **floating-point number regression** tasks [38, 52]. Hafner u.a. [52] reported that **using two-hot encoding** significantly improved the prediction of rewards and values in reinforcement learning.

The exceptional scalability of Large Language Models (LLMs) [36, 39] with model size can likely be attributed to two factors: the inherently discrete nature of text data and the widespread use of the Adam optimizer. As LLMs continue to evolve into foundation models [45] for various modalities, image, speech, and video domains are also adopting discrete token representations [49, 51, 57]. This provides further evidence that empirical FIM estimation necessitates the use of discrete distributions.

3.4 Fisher Adam

Based on the discussion in this section, we need to make minimal modifications to the Adam algorithm. We call this **FAdam (Fisher Adam)**, and it is presented in Algorithm 1.

Algorithm 1 Fisher Adam (FAdam)

1: **given** $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, $\epsilon_2 = 0.01$, $c = 1$, $\lambda = 0.001$, $\rho = 0.5$, η_t
2: **initialize** $\boldsymbol{\theta}_0$, $t \leftarrow 0$, $\boldsymbol{m}_0 \leftarrow 0_N$, $\boldsymbol{f}_0 \leftarrow 1_N$ ▷ FIM init to 1 as per Section 3.4.4
3: **repeat**
4:     $t \leftarrow t + 1$
5:     $\boldsymbol{g}_t \leftarrow \nabla_{\boldsymbol{\theta}} \log P_t(\boldsymbol{\theta}_{t-1})$ ▷ Stochastic gradient as per Eq. (12)
6:     $\hat{\beta}_2 \leftarrow \beta_2(1 - \beta_2^{t-1})/(1 - \beta_2^t)$ ▷ Bias correction as per Section 3.4.4
7:     $\boldsymbol{f}_t \leftarrow \hat{\beta}_2 \boldsymbol{f}_{t-1} + (1 - \hat{\beta}_2)\boldsymbol{g}_t^2$ ▷ EMA diagonal empirical FIM as per Section 3.4.1
8:     $\hat{\epsilon} \leftarrow \min(\epsilon, \epsilon_2 \text{RMS}(\boldsymbol{g}_t))$ ▷ Adaptive epsilon as per Appendix B.3
9:     $\bar{\boldsymbol{g}}_t \leftarrow \boldsymbol{g}_t/(\boldsymbol{f}_t^{\rho} + \hat{\epsilon}^{2\rho})$ ▷ Invariant natural gradient as per Eq. (27)
10:     $\bar{\boldsymbol{g}}_t \leftarrow \bar{\boldsymbol{g}}_t/\max(1, \text{RMS}(\bar{\boldsymbol{g}}_t)/c)$ ▷ Clip the gradient as per Appendix B.3
11:     $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\bar{\boldsymbol{g}}_t$ ▷ EMA momentum as per Section 3.4.2
12:     $\bar{\boldsymbol{g}}_w \leftarrow \boldsymbol{\theta}_{t-1}/(\boldsymbol{f}_t^{\rho} + \hat{\epsilon}^{2\rho})$ ▷ Weight decay as per Eq. (28)
13:     $\bar{\boldsymbol{g}}_w \leftarrow \bar{\boldsymbol{g}}_w/\max(1, \text{RMS}(\bar{\boldsymbol{g}}_w)/c)$ ▷ Clip weight decay as per Appendix B.3
14:     $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_t(\boldsymbol{m}_t + \lambda \bar{\boldsymbol{g}}_w)$ ▷ Update $\boldsymbol{\theta}$ as per Eq. (13)
15: **until** *stopping criterion is met*
16: **return** optimized parameters $\boldsymbol{\theta}_t$

3.4.1 Reciprocal vs Reciprocal Square-root for FIM

We have come to realize that the second momentum term in Adam [12] is actually a diagonal empirical FIM. However, in contrast to Eq. (22) and Eq. (59), where gradients are divided by FIM, Adam divides gradients by the square root of FIM, as shown Algorithm 3.

To precisely update $\boldsymbol{\theta}_t$ at each training step as per Eq. (13), we ideally need an accurate estimation of FIM. However, the empirical FIM computed by the minibatch data $\mathcal{B}$ is noisy. Relying on a diagonal empirical FIM can result in zero components, which causes the natural gradient to diverge due to division by zero. To address this and obtain a more stable diagonal empirical FIM, an exponential moving average (EMA) is employed.

The gradient and FIM in Eq. (22) vary at every point $\boldsymbol{\theta}$. As shown in Eq. (1), not only the components of the gradient and FIM change but also their basis. However, EMA averages the components of

---

[4]Diffusion models [32] seem to perform well with L2 loss. It is likely because the expectation of the loss is taken over not only image samples but also the noisy diffusion steps, which cover the model distribution sufficiently, allowing for a more accurate empirical estimate of the FIM. This could be a significant factor contributing to the success of diffusion models.

the Fisher information tensor obtained across different $\boldsymbol{\theta}$ points, even though the basis for each FIM might be different. This is a mathematically invalid operation.[5] As β_2=0.999 is standard value in Adam, EMA has a half-life of 700 steps ($\sim \log(0.5)/\log(0.999)$), meaning it averages FIM over roughly 1000 steps. During this time, the basis can shift significantly as the model parameter moves to a new $\boldsymbol{\theta}$ coordinate on the manifold.

In differential geometry, scalars, vectors and tensors are invariant under a coordinate change. The magnitude of a vector, derived from the dot product, is a scalar. For a given gradient, we can express its magnitude as shown in Eq. (10), and simplify it using the diagonal FIM in Eq. (26). As Eq. (26) is a scalar, it can be subjected to EMA. We utilize this value for the natural gradient and refer to it as the **invariant natural gradient**, denoted as $\bar{\nabla} J(\boldsymbol{\theta})$ in Eq. (27). This is **the reason why Adam uses the square root**, as shown in Algorithm 3. In section C.1.1, we conduct an ablation study on the exponent of the FIM term, and find that the square root is the optimal choice.

$$\|\tilde{\nabla} J(\boldsymbol{\theta})\|^2 = \tilde{\nabla} J(\boldsymbol{\theta}) \cdot \tilde{\nabla} J(\boldsymbol{\theta}) = (\boldsymbol{F}^{-1} \nabla J(\boldsymbol{\theta}))^\top \boldsymbol{F} (\boldsymbol{F}^{-1} \nabla J(\boldsymbol{\theta})) = \nabla J(\boldsymbol{\theta})^\top \boldsymbol{F}^{-1} \nabla J(\boldsymbol{\theta}) \quad (25)$$

$$\approx \nabla J(\boldsymbol{\theta})^\top \hat{\boldsymbol{F}}^{-1} \nabla J(\boldsymbol{\theta}) = \|\nabla J(\boldsymbol{\theta})/\sqrt{\hat{\boldsymbol{F}}}\|^2 \quad (26)$$

$$\bar{\nabla} J(\boldsymbol{\theta}) := \nabla J(\boldsymbol{\theta})/\sqrt{\hat{\boldsymbol{F}}} \quad (27)$$

Adam variants like AdaDelta [7], AdaMax [12] and Yogi [21] modify the second momentum to further deviate from the natural gradient. A more principled approach would focus on improving the estimation of the empirical FIM, incorporating off-diagonal elements, and investigating how to make using natural gradient, instead of invariant natural gradient. If we were able to achieve a more accurate FIM estimation with smaller β_2, it might be possible to eliminate the square root operation.

3.4.2 Momentum

Momentum methods [2] employ exponential moving average (EMA) of gradients along the trajectory because the stochastic gradients estimated from the minibatch $\mathcal{B}$ are noisy.

At point $\boldsymbol{\theta}$ on the statistical manifold, the gradient is the invariant natural gradient, as shown in Eq. (27). Therefore, momentum should average the invariant natural gradient $\bar{\nabla} J(\boldsymbol{\theta})$ instead of the gradient, as shown in Algorithm 1. In Algorithm 3, Adam [12] calculates the first and second momentums separately and then combines them, which lacks a theoretical background. In Algorithm 4, Adafactor [20], on the other hand, already follows our proposed approach. Furthermore, the second momentum is not momentum. We refer to it as FIM.

FAdam omits zero bias correction in EMA of momentum [6], because excessive gradient descent is unnecessary before the momentum is sufficiently accumulated. This approach can be viewed as built-in informed warmup schedule. Adafactor implementation [68] already omits zero bias correction for momentum, although this is not explicitly mentioned in the paper [20].

3.4.3 Weight decay on manifold

AdamW [16], which decouples the weight decay term from the loss and applies it directly to the Adam optimizer, has demonstrated general performance improvements and is now widely used as a standard practice. Our mathematical framework provides a sound theoretical explanation for this observed phenomenon. The loss function should be log-likelihood, but the weight decay term has nothing to do with the probability distribution. If the weight decay term is included in the loss, it causes problems in estimating the FIM.

Weight decay is a great example of how auxiliary losses should be handled. **If the auxiliary loss is related to the log-likelihood, it can be included in the loss. Otherwise, it should be bypassed by the Adam optimizer, as in the case of weight decay**.

---

[5]To add tensors from different tangent spaces, they must be parallel transported to the desired tangent space. This requires knowledge of how the manifold's basis changes over the manifold, which is determined by the Riemannian metric tensor field. However, accurately knowing the FIM even at a single point is computationally prohibitive, making knowledge of the FIM tensor field intractable.

[6]$\boldsymbol{m}_t \leftarrow \boldsymbol{m}_t/(1 - \beta_1^t)$

Furthermore, following Eq. (11), weight decay should also be applied as a natural gradient. Since we already have the FIM, we can express the weight decay gradient in a similar way to Eq. (27).

$$\boldsymbol{g}_w = \boldsymbol{\theta}/\sqrt{\hat{\boldsymbol{F}}} \tag{28}$$

Intuitively, components with low Fisher information can be pushed closer to zero without significantly impacting the model performance. Elastic weight consolidation [19] leverages a similar concept, utilizing Fisher information to regularize the change of $\boldsymbol{\theta}$.[7]

It has been reported that for training large-scale models like LLMs, decoupling weight decay from the learning rate helps stabilize training [55]. However, since our modifications make the weight decay adaptable to the loss surface, such workarounds might not be necessary. Further research is needed to confirm this hypothesis.

3.4.4 FAdam

Fisher Adam (FAdam), incorporating all the discussed modifications, is presented in Algorithm 1. Fisher Adafactor (FAdafactor) modification of Adafactor is presented in Algorithm 2 in Appendix B.4.

Additionally, we have specified that FIM ($\boldsymbol{f}_0$) is initialized to 1, not 0, in Algorithm 1. This is because FIM represents a Riemannian metric, which defaults to the identity matrix in flat space. However, this change does not affect the logic of the algorithm because bias correction ignores the initial value. The bias correction for FIM is adopted from Adafactor [20], which is agnostic to the initial value.

3.4.5 Convergence analysis

Algorithm 1 does not deviate from the assumptions of the convergence analysis presented in Eq. (15) of the latest Adam convergence proof by Défossez u.a. [40]. Therefore, FAdam achieves the following convergence guarantees. Specifically, as the number of iterations N becomes sufficiently large, the expected value of the gradient becomes sufficiently small. The detailed explanation is provided in Appendix B.6.

$$\mathbb{E}\left[||\nabla F(x_\tau)||^2\right] \leq O(d\ln(N)/\sqrt{N}) \tag{29}$$

4 Experiment

While β_2, λ and η_t typically require tuning for each specific model in Algorithm 1, the remaining hyperparameters performed well across different domains and model sizes in our experiments.

4.1 LLM (text domain)

We pretrained the 1B parameter LLM model from PaLM [54] on C4 dataset [34]. The hyperparameters used for FAdafactor were β_1=0.9, β_2=0.99, ϵ=1e-8, and λ=0.1. In contrast, the Adafactor hyperparameters were β_1=0.9, β_2=0.99, ϵ=1e-30 and λ=0.001. As demonstrated in Fig. 1a, FAdafactor outperforms Adafactor.

4.2 ASR (speech domain)

The 600M parameter Conformer [31] model from the w2v-BERT [47] is one of the lowest WER (word error rate) achieving models on the LibriSpeech dataset [14]. This model was pretrained using w2v-BERT and then finetuned on LibriSpeech data using the RNNT loss [9]. The WER was further improved to SoTA levels using noisy student semi-supervised finetuning [35] on LibriLight [37]. We compared Adam and FAdam using models that were fine-tuned in a semi-supervised fashion on LibriLight data, paired with semi-supervised pseudo labels.

FAdam and Adam have identical hyperparameters as follows: β_1=0.9, β_2=0.98, ϵ=1e-8, and λ=0.001. As demonstrated in Fig. 1b and Table 1, FAdam not only outperforms Adam but also establishes a new state-of-the-art (SoTA) Word Error Rate (WER) on LibriSpeech for 600M parameter models.

---

[7]EWC can directly utilize FIM from FAdam, eliminating the need to compute it separately.

(a) LLM 1B (text domain)

(b) ASR 600M (speech domain)

(c) VQ-VAE 100M (image domain)
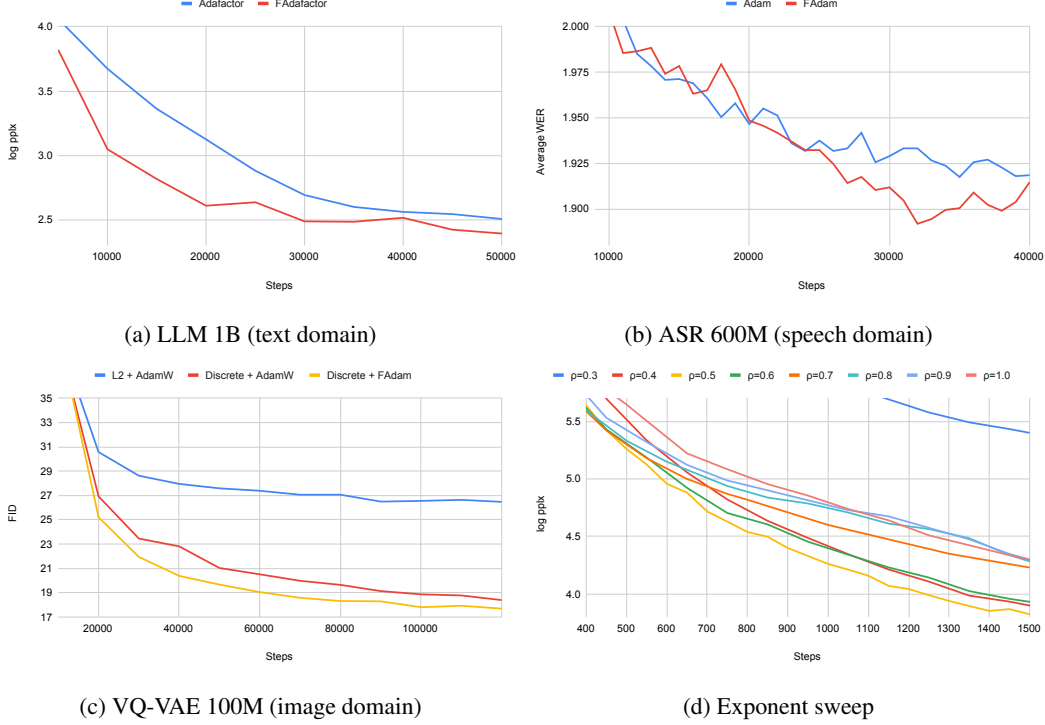

(d) Exponent sweep

Figure 1: Comparison of FAdam and Adam performance. (a) Eval loss (log pplx) on 1B LLMs, presenting FAdafactor outperforms Adafactor. (b) Average WER on LibriSpeech using 600M Conformer models, presenting FAdam outperforms Adam. (c) FID of ImageNet generation using 100M VQ-VAE models, presenting FAdam outperforms AdamW. (d) Comparison of FIM exponents on a 1B LLM, showing 0.5 (square root) as the optimal choice.

LibriSpeech WERs	dev	dev-other	test	test-other	avg
Adam (w2v-BERT paper [47])	1.30	2.60	1.40	2.70	2.00
Adam	1.30	2.54	1.33	2.59	1.93
FAdam	1.29	2.49	1.34	2.49	**1.89**

Table 1: LibriSpeech WERs

4.3 VQ-VAE (image domain)

We trained a 100M parameter ViT VQ-GAN model [46] on the ImageNet dataset [13]. To verify the hypothesis from Section 3.3.1 that categorical cross-entropy (CE) loss is superior to L2 loss on Adam, we exclusively used either CE loss or L2 loss + logit-laplace during training.[8] The VQ-GAN paper [46] introduced logit-laplace to adjust the output scale when using L2 loss.

FAdam and AdamW [16] have identical hyperparameters as follows: β_1=0.9, β_2=0.99, ϵ=1e-8, and λ=1e-4. As demonstrated in Fig. 1c, FAdam outperforms AdamW, and categorical CE loss not only outperforms L2 loss but also eliminates the need for the complex logit-laplace transformation.

4.4 Ablation study

We conducted ablation studies on each hyperparameter of FAdam. Due to page limitations in the main paper, the results are presented in Appendix C.1.

---

[8]Since our primary focus is on replacing the L2 loss, we omitted GAN and perceptual losses, making the model effectively a VQ-VAE [18] rather than a VQ-GAN.

5 Conclusion

In this work, we have revealed the mathematical foundation of the Adam optimizer, clarifying the constraints on loss function selection and the approximations involved in its derivation. This groundwork opens avenues for future research to develop improved optimizers by mitigating these approximations. Building upon our theoretical foundation, we have proposed an enhanced algorithm, FAdam, and demonstrated its effectiveness across diverse domains.

Acknowledgement

References

[1] Parisi, Giorgio(1981): *Correlation functions and computer simulations*, 3: 378–384.

[2] Rumelhart, David E / Hinton, Geoffrey E / Williams, Ronald J(1986): *Learning representations by back-propagating errors*, 6088: 533–536.

[3] Amari, Shun Ichi(1998): *Natural gradient works efficiently in learning*, 2: 251–276.

[4] Schraudolph, Nicol N(2002): *Fast curvature matrix-vector products for second-order gradient descent*, 7: 1723–1738.

[5] Martens, James / others u.a.(2010): *Deep learning via hessian-free optimization.*In: ICML735–742.

[6] Neal, Radford M / others u.a.(2011): *MCMC using Hamiltonian dynamics*, 11: 2.

[7] Zeiler, Matthew D(2012): *Adadelta: an adaptive learning rate method*.

[8] Vinyals, Oriol / Povey, Daniel(2012): *Krylov subspace descent for deep learning*In: Artificial intelligence and statistics1261–1268.

[9] Graves, Alex(2012): *Sequence transduction with recurrent neural networks*.

[10] Pascanu, Razvan / Mikolov, Tomas / Bengio, Yoshua(2013): *On the difficulty of training recurrent neural networks*In: International conference on machine learning1310–1318.

[11] Pascanu, Razvan / Bengio, Yoshua(2013): *Revisiting natural gradient for deep networks*.

[12] Kingma, Diederik P / Ba, Jimmy(2014): *Adam: A method for stochastic optimization*.

[13] Russakovsky, Olga u.a.(2015): *Imagenet large scale visual recognition challenge*211–252.

[14] Panayotov, Vassil / Chen, Guoguo / Povey, Daniel / Khudanpur, Sanjeev(2015): *Librispeech: an asr corpus based on public domain audio books*In: 2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)5206–5210.

[15] Martens, James / Grosse, Roger(2015): *Optimizing neural networks with kronecker-factored approximate curvature*In: International conference on machine learning2408–2417.

[16] Loshchilov, Ilya / Hutter, Frank(2017): *Decoupled weight decay regularization*.

[17] Smith, Samuel L / Kindermans, Pieter Jan / Ying, Chris / Le, Quoc V(2017): *Don't decay the learning rate, increase the batch size*.

---

[9]`https://github.com/lessw2020/FAdam_PyTorch`

[18] Van Den Oord, Aaron / Vinyals, Oriol / others u.a.(2017): *Neural discrete representation learning*.

[19] Kirkpatrick, James u.a.(2017): *Overcoming catastrophic forgetting in neural networks*, 13: 3521–3526.

[20] Shazeer, Noam / Stern, Mitchell(2018): *Adafactor: Adaptive learning rates with sublinear memory cost*In: International Conference on Machine Learning4596–4604.

[21] Zaheer, Manzil / Reddi, Sashank / Sachan, Devendra / Kale, Satyen / Kumar, Sanjiv(2018): *Adaptive methods for nonconvex optimization*.

[22] Mahalanobis, Prasanta Chandra(2018): *On the generalized distance in statistics*S1–S7.

[23] Bottou, Léon / Curtis, Frank E / Nocedal, Jorge(2018): *Optimization methods for large-scale machine learning*, 2: 223–311.

[24] Gupta, Vineet / Koren, Tomer / Singer, Yoram(2018): *Shampoo: Preconditioned stochastic tensor optimization*In: International Conference on Machine Learning1842–1850.

[25] Lydia, Agnes / Francis, Sagayaraj(2019): *Adagrad—an optimizer for stochastic gradient descent*, 5: 566–568.

[26] Luo, Liangchen / Xiong, Yuanhao / Liu, Yan / Sun, Xu(2019): *Adaptive gradient methods with dynamic bound of learning rate*.

[27] Amari, Shun ichi / Karakida, Ryo / Oizumi, Masafumi(2019): *Fisher information and natural gradient learning in random deep networks*In: The 22nd International Conference on Artificial Intelligence and Statistics694–702.

[28] Kunstner, Frederik / Hennig, Philipp / Balles, Lukas(2019): *Limitations of the empirical fisher approximation for natural gradient descent*.

[29] Anil, Rohan / Gupta, Vineet / Koren, Tomer / Singer, Yoram(2019): *Memory efficient adaptive optimization*.

[30] Reddi, Sashank J / Kale, Satyen / Kumar, Sanjiv(2019): *On the convergence of adam and beyond*.

[31] Gulati, Anmol u.a.(2020): *Conformer: Convolution-augmented transformer for speech recognition*.

[32] Ho, Jonathan / Jain, Ajay / Abbeel, Pieter(2020): *Denoising diffusion probabilistic models*6840–6851.

[33] Yuan, Wei / Gao, Kai Xin(2020): *EAdam Optimizer: How ϵ Impact Adam*.

[34] Raffel, Colin u.a.(2020): *Exploring the limits of transfer learning with a unified text-to-text transformer*, 140: 1–67.

[35] Park, Daniel S / Zhang, Yu / Jia, Ye / Han, Wei / Chiu, Chung Cheng / Li, Bo / Wu, Yonghui / Le, Quoc V(2020): *Improved noisy student training for automatic speech recognition*.

[36] Brown, Tom u.a.(2020): *Language models are few-shot learners*1877–1901.

[37] Kahn, Jacob u.a.(2020): *Libri-light: A benchmark for asr with limited or no supervision*In: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)7669–7673.

[38] Sønderby, Casper Kaae u.a.(2020): *Metnet: A neural weather model for precipitation forecasting*.

[39] Kaplan, Jared u.a.(2020): *Scaling laws for neural language models*.

[40] Défossez, Alexandre / Bottou, Léon / Bach, Francis / Usunier, Nicolas(2020): *A simple convergence proof of adam and adagrad*.

[41] Zhang, Jingzhao / Karimireddy, Sai Praneeth / Veit, Andreas / Kim, Seungyeon / Reddi, Sashank / Kumar, Sanjiv / Sra, Suvrit(2020): *Why are adaptive methods good for attention models?*15383–15393.

[42] Jelassi, Samy / Mensch, Arthur / Gidel, Gauthier / Li, Yuanzhi(2021): *Adam is no better than normalized SGD: Dissecting how adaptivity improves GAN performance*.

[43] Bengio, Yoshua / Lahlou, Salem / Deleu, Tristan / Hu, Edward J / Tiwari, Mo / Bengio, Emmanuel(2021): *Gflownet foundations*.

[44] Gilmer, Justin u.a.(2021): *A loss curvature perspective on training instability in deep learning*.

[45] Bommasani, Rishi u.a.(2021): *On the opportunities and risks of foundation models*.

[46] Yu, Jiahui u.a.(2021): *Vector-quantized image modeling with improved vqgan*.

[47] Chung, Yu An / Zhang, Yu / Han, Wei / Chiu, Chung Cheng / Qin, James / Pang, Ruoming / Wu, Yonghui(2021): *W2v-bert: Combining contrastive learning and masked language modeling for self-supervised speech pre-training*In: 2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)244–250.

[48] Cohen, Jeremy M u.a.(2022): *Adaptive gradient methods at the edge of stability*.

[49] Borsos, Zalán u.a.(2023): *Audiolm: a language modeling approach to audio generation*.

[50] Berman, David S / Klinger, Marc S / Stapleton, Alexander G(2023): *Bayesian renormalization*, 4: 045011.

[51] Team, Gemini u.a.(2023): *Gemini: a family of highly capable multimodal models*.

[52] Hafner, Danijar / Pasukonis, Jurgis / Ba, Jimmy / Lillicrap, Timothy(2023): *Mastering diverse domains through world models*.

[53] Kunstner, Frederik / Chen, Jacques / Lavington, Jonathan Wilder / Schmidt, Mark(2023): *Noise is not the main factor behind the gap between sgd and adam on transformers, but sign descent might be*.

[54] Chowdhery, Aakanksha u.a.(2023): *Palm: Scaling language modeling with pathways*, 240: 1–113.

[55] Wortsman, Mitchell u.a.(2023): *Small-scale proxies for large-scale transformer training instabilities*.

[56] Jouppi, Norm u.a.(2023): *Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings*In: Proceedings of the 50th Annual International Symposium on Computer Architecture1–14.

[57] Lu, Jiasen / Clark, Christopher / Lee, Sangho / Zhang, Zichen / Khosla, Savya / Marten, Ryan / Hoiem, Derek / Kembhavi, Aniruddha(2023): *Unified-io 2: Scaling autoregressive multimodal models with vision, language, audio, and action*.

[58] Martins Gomes, Damien / Zhang, Yanlei / Belilovsky, Eugene / Wolf, Guy / Hosseini, Mahdi S(2024): *AdaFisher: Adaptive Second Order Optimization via Fisher Information*arXiv–2405.

[59] Wang, Xi / Aitchison, Laurence(2024): *Batch size invariant Adam*.

[60] Lin, Wu / Dangel, Felix / Eschenhagen, Runa / Bae, Juhan / Turner, Richard E / Makhzani, Alireza(2024): *Can We Remove the Square-Root in Adaptive Gradient Methods? A Second-Order Perspective*.

[61] Berman, David S / Klinger, Marc S(2024): *The inverse of exact renormalization group flows as statistical inference*, 5: 389.

[62] Mukherjee, Anirbit / Tucat, Matteo(2024): *Regularized Gradient Clipping Provably Trains Wide and Deep Neural Networks*.

[63] Amari, Shun ichi (2012): *Differential-geometrical methods in statistics*. , Springer Science & Business Media.

[64] Berman, David S / Heckman, Jonathan J / Klinger, Marc (2023): *On the dynamics of inference and learning*. MACHINE LEARNING: IN PURE MATHEMATICS AND THEORETICAL PHYSICS.World Scientific: 41–83.

[65] Bonnans, Joseph Frédéric / Gilbert, Jean Charles / Lemaréchal, Claude / Sagastizábal, Claudia A (2006): *Numerical optimization: theoretical and practical aspects*. , Springer Science & Business Media.

[66] Kristiadi, Agustinus (2018): *Natural Gradient Descent*
`https://agustinus.kristia.de/techblog/2018/03/14/natural-gradient/`.

[67] Petersen, Peter (2006): *Riemannian geometry*. , Springer.

[68] Shazeer, Noam (2018): *Adafactor implementation*
`https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/utils/adafactor.py`.

A   Background

A.1   Tensor Calculus Notation

In this paper, we will derive the Adam algorithm from the perspective of information geometry. Since statistical formulas are frequently used in machine learning papers, but Riemannian geometry formulas are not, we will first clarify the notation before proceeding. We follow the notation of Petersen [67]'s Riemannian geometry textbook.

We define the tangent space at a point $\boldsymbol{\theta}$ on a manifold as $\mathcal{T}_{\boldsymbol{\theta}}\mathcal{M}$. Given a vector $\vec{v} = v^i \boldsymbol{e}_i$ on $\mathcal{T}_{\boldsymbol{\theta}}\mathcal{M}$, the basis vectors are defined as partial derivatives as follows:

$$\boldsymbol{e}_i := \frac{\partial}{\partial \theta^i} := \partial_i \tag{30}$$

Therefore, a vector $\vec{v}$ can be expressed using Einstein notation as follows, with v^i representing the vector components and ∂_i representing the basis vectors:

$$\vec{v} = \boldsymbol{v} = v^i \partial_i \tag{31}$$

Given a covector $\boldsymbol{w} = w_i \boldsymbol{e}^i$ in the dual space of a vector space, the covector basis is defined using a differential one-form.

$$\boldsymbol{e}^i := d\theta^i := d^i \tag{32}$$

The operation between the vector basis and the covector basis is the Kronecker delta. The operation between vector $\vec{v}$ and covector $\boldsymbol{w}$ is as follows:

$$d^i(\partial_j) = \delta^i_j \tag{33}$$

$$\boldsymbol{w}(\vec{v}) = v^i w_j d^i(\partial_j) = v^i w_i \tag{34}$$

The covariant derivative is an extension of the directional derivative operation to vectors and tensors. When applied to a scalar field (i.e., a scalar function), the covariant derivative is called the directional derivative. The covariant derivative of a vector field $\vec{w}$ along a vector $\vec{v}$ is denoted as follows, where Γ^k_{ij} represents the Christoffel symbols:

$$D_{\vec{v}}\vec{w} = v^i \partial_i(w^j \partial_j) = v^i \partial_i w^j \partial_j + v^i w^j \partial_i \partial_j = (v^i \partial_i w^k + v^i w^j \Gamma^k_{ij})\partial_k \tag{35}$$

A.2   The proof of Fisher is negative Hessian

The relationship between Fisher information and the negative Hessian's expectation can be proven as follows:

$$\nabla^2_{\boldsymbol{\theta}} \log P(\mathrm{x}|\theta) = \frac{\nabla^2_{\boldsymbol{\theta}} P(\mathrm{x}|\theta)}{P(\mathrm{x}|\theta)} - \nabla_{\boldsymbol{\theta}} \log P(\mathrm{x}|\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log P(\mathrm{x}|\boldsymbol{\theta})^\top \tag{36}$$

$$\mathbb{E}_{\mathrm{x} \sim P_{\boldsymbol{\theta}}} \left[\frac{\nabla^2_{\boldsymbol{\theta}} P(\mathrm{x}|\theta)}{P(\mathrm{x}|\theta)} \right] = \int P(\mathrm{x}|\theta) \frac{\nabla^2_{\boldsymbol{\theta}} P(\mathrm{x}|\theta)}{P(\mathrm{x}|\theta)} d\mathrm{x} = \nabla^2_{\boldsymbol{\theta}} \int P(\mathrm{x}|\theta) d\mathrm{x} = \nabla^2_{\boldsymbol{\theta}} 1 = 0 \tag{37}$$

$$\mathbb{E}_{\mathrm{x} \sim P_{\boldsymbol{\theta}}} \left[\nabla^2_{\boldsymbol{\theta}} \log P(\mathrm{x}|\theta) \right] = -\mathbb{E}_{\mathrm{x} \sim P_{\boldsymbol{\theta}}} \left[\nabla_{\boldsymbol{\theta}} \log P(\mathrm{x}|\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log P(\mathrm{x}|\boldsymbol{\theta})^\top \right] \tag{38}$$

A.3   The proof of KL approximation

As seen in Eq. (9), the Kullback-Leibler (KL) divergence with an infinitesimal displacement $\boldsymbol{d}$ can be approximated by the Fisher information. We provide a proof of this relationship below, following the approach presented in Kristiadi [66].

$$D_{\mathrm{KL}}(P(\mathrm{x}|\boldsymbol{\theta})\|P(\mathrm{x}|\boldsymbol{\theta}+\boldsymbol{d})) \approx \frac{1}{2}\boldsymbol{d}^\top \boldsymbol{F}(\boldsymbol{\theta})\boldsymbol{d}. \tag{39}$$

By using a second-order Taylor series approximation, the KL divergence can be approximated as follows:

$$D_{\mathrm{KL}}(P_{\boldsymbol{\theta}}\|P_{\boldsymbol{\theta}+\boldsymbol{d}}) \approx D_{\mathrm{KL}}(P_{\boldsymbol{\theta}}\|P_{\boldsymbol{\theta}}) + (\nabla_{\boldsymbol{\theta}'}D_{\mathrm{KL}}(P_{\boldsymbol{\theta}}\|P_{\boldsymbol{\theta}'})|_{\boldsymbol{\theta}=\boldsymbol{\theta}'})^\top \boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^\top \nabla_{\boldsymbol{\theta}'}^2 D_{\mathrm{KL}}(P_{\boldsymbol{\theta}}\|P_{\boldsymbol{\theta}'})\boldsymbol{d} \tag{40}$$

$D_{\mathrm{KL}}(P_{\boldsymbol{\theta}}\|P_{\boldsymbol{\theta}})$ is 0 by the definition of KL divergence. The first-order approximation term also becomes 0 through the following process.

$$\nabla_{\boldsymbol{\theta}'}D_{\mathrm{KL}}(P_{\boldsymbol{\theta}}\|P_{\boldsymbol{\theta}'}) = \cancel{\nabla_{\boldsymbol{\theta}'}\mathbb{E}_{\mathrm{x}\sim P_{\boldsymbol{\theta}}}[\log P_{\boldsymbol{\theta}}]} - \nabla_{\boldsymbol{\theta}'}\mathbb{E}_{\mathrm{x}\sim P_{\boldsymbol{\theta}}}[\log P_{\boldsymbol{\theta}'}] = -\mathbb{E}_{\mathrm{x}\sim P_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}'}\log P_{\boldsymbol{\theta}'}] \tag{41}$$

$$= -\int P_{\boldsymbol{\theta}}\nabla_{\boldsymbol{\theta}'}\log P_{\boldsymbol{\theta}'}|_{\boldsymbol{\theta}=\boldsymbol{\theta}'}d\mathrm{x} = -\int P_{\boldsymbol{\theta}}\frac{\nabla_{\boldsymbol{\theta}'}P_{\boldsymbol{\theta}'}}{P_{\boldsymbol{\theta}'}}|_{\boldsymbol{\theta}=\boldsymbol{\theta}'}d\mathrm{x} \tag{42}$$

$$= -\int \nabla_{\boldsymbol{\theta}}P_{\boldsymbol{\theta}}d\mathrm{x} = -\nabla_{\boldsymbol{\theta}}\int P_{\boldsymbol{\theta}}d\mathrm{x} = -\nabla_{\boldsymbol{\theta}}1 = 0 \tag{43}$$

The Fisher information emerges from the second-order approximation term, as shown by utilizing the Hessian property in Eq. (8).

$$\nabla_{\boldsymbol{\theta}'}^2 D_{\mathrm{KL}}(P_{\boldsymbol{\theta}}\|P_{\boldsymbol{\theta}'}) = \cancel{\nabla_{\boldsymbol{\theta}'}^2\mathbb{E}_{\mathrm{x}\sim P_{\boldsymbol{\theta}}}[\log P_{\boldsymbol{\theta}}]} - \nabla_{\boldsymbol{\theta}'}^2\mathbb{E}_{\mathrm{x}\sim P_{\boldsymbol{\theta}}}[\log P_{\boldsymbol{\theta}'}] \tag{44}$$

$$= -\mathbb{E}_{\mathrm{x}\sim P_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}'}^2\log P_{\boldsymbol{\theta}'}|_{\boldsymbol{\theta}=\boldsymbol{\theta}'}] = -\mathbb{E}_{\mathrm{x}\sim P_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}}^2\log P_{\boldsymbol{\theta}}] \tag{45}$$

$$= \boldsymbol{F}(\boldsymbol{\theta}) \tag{46}$$

Therefore, equation Eq. (9) is proven.

B Toward FAdam

B.1 Compared to Newton's second-order method

As shown in Eq. (8), FIM is equivalent to the Hessian of the loss.

$$\boldsymbol{F}(\boldsymbol{\theta}) = -\mathbb{E}_{\mathrm{x}\sim P_{\boldsymbol{\theta}}}[\boldsymbol{H}_{\boldsymbol{\theta}}(\log P(\mathrm{x}|\boldsymbol{\theta}))] = \mathbb{E}_{\mathrm{x}\sim P_{\boldsymbol{\theta}}}[\boldsymbol{H}_{\boldsymbol{\theta}}(\mathcal{L}(\boldsymbol{\theta}))]. \tag{47}$$

Therefore, $\boldsymbol{\theta}$ update Eq. (13) incorporates the Hessian term.

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\mathbb{E}_{\mathrm{x}\sim P_{\boldsymbol{\theta}}}[\boldsymbol{H}_{\boldsymbol{\theta}}(\mathcal{L}(\boldsymbol{\theta}))]^{-1}\nabla\mathcal{L}(\boldsymbol{\theta}) \tag{48}$$

As FIM is the expected value of the Hessian, natural gradient optimization is considered a second order method. Since the Fisher Information Matrix (FIM) is positive semi-definite, the Hessian term is also positive semi-definite, ensuring a convex optimization.

Meanwhile, Newton's method is a second-order optimization method that is effective only when the loss function is strongly convex with a Lipschitz continuous Hessian. The update equation for Newton's method [65] is as follows.

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\boldsymbol{H}_{\boldsymbol{\theta}}(\mathcal{L}(\boldsymbol{\theta}))^{-1}\nabla\mathcal{L}(\boldsymbol{\theta}) \tag{49}$$

Although the two methods were derived through vastly different processes, they both involve the inverse of the Hessian matrix, as shown in Eq. (48) and Eq. (49). However, there is a significant difference between the two methods. Natural gradient descent requires the loss function to be the log-likelihood, while Newton's method has no such restriction on the choice of loss function. Natural

gradient descent guarantees convergence due to the positive semi-definiteness of FIM (the expected value of the Hessian), whereas Newton's method does not.

Consequently, various complex techniques are employed when using Newton's method, such as the Gauss-Newton algorithm, conjugate gradient method, and trust region method [65]. These techniques ensure that each step update is confined within a trust region, mitigating the risk of divergence.

This is why second-order optimization methods based on FIM have demonstrated faster convergence compared to Newton's method in practice [4, 5, 8]. Recent second-order optimization methods often utilize FIM instead of the Hessian of the loss function [24, 29]. However, these methods require the loss function to be the log-likelihood and are subject to certain constraints discussed in Section 3.2 and Section 3.3.

B.2   Conditional probability distribution for empirical Fisher Information

In supervised learning, the loss function typically becomes a joint distribution $-\log P(\mathrm{x}, \mathrm{y}|\boldsymbol{\theta})$. Since the expected value over $P(\mathrm{x}, \boldsymbol{\theta})$ is generally intractable, it is replaced with the empirical distribution $\hat{p}_{data}(\mathrm{x})$ in Eq. (51). As noted in Eq. (16), this is referred to as the empirical Fisher. The training set also provides the ground truth label y. Therefore, the loss function becomes the cross-entropy calculated using the conditional log-likelihood, as shown in Eq. (52).

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = -\mathbb{E}_{\mathrm{x,y}\sim p_{data}}[\nabla_{\boldsymbol{\theta}} \log P(\mathrm{y}|\mathrm{x}, \boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}} \log P(\mathrm{x}|\boldsymbol{\theta})] \tag{50}$$

$$\approx -\mathbb{E}_{\mathrm{x,y}\sim \hat{p}_{data}}[\nabla_{\boldsymbol{\theta}} \log P(\mathrm{y}|\mathrm{x}, \boldsymbol{\theta}) + \underline{\nabla_{\boldsymbol{\theta}} \log \hat{p}_{data}(\mathrm{x})}] \tag{51}$$

$$= -\mathbb{E}_{\mathrm{x,y}\sim \hat{p}_{data}}[\nabla_{\boldsymbol{\theta}} \log P(\mathrm{y}|\mathrm{x}, \boldsymbol{\theta})] := \boldsymbol{g} \tag{52}$$

The FIM, similar to the cost function, also requires the calculation of an expected value over the joint distribution. By approximating the expectation with the empirical distribution $\hat{p}_{data}(\mathrm{x}, \mathrm{y})$ and moving out the square as shown in Eq. (21), we obtain the following equation:

$$\hat{\boldsymbol{F}}(\boldsymbol{\theta}) = \mathbb{E}_{\mathrm{x,y}\sim P(\mathrm{y}|\mathrm{x},\boldsymbol{\theta})P(\mathrm{x}|\boldsymbol{\theta})}[(\nabla_{\boldsymbol{\theta}} \log P(\mathrm{y}|\mathrm{x}, \boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}} \log P(\mathrm{x}|\boldsymbol{\theta}))^2] \tag{53}$$

$$\approx \mathbb{E}_{\mathrm{x,y}\sim P(\mathrm{y}|\mathrm{x},\boldsymbol{\theta})\hat{p}_{data}(\mathrm{x})}[(\nabla_{\boldsymbol{\theta}} \log P(\mathrm{y}|\mathrm{x}, \boldsymbol{\theta}) + \underline{\nabla_{\boldsymbol{\theta}} \log \hat{p}_{data}(\mathrm{x})})^2] \tag{54}$$

$$\approx \mathbb{E}_{\mathrm{x,y}\sim P(\mathrm{y}|\mathrm{x},\boldsymbol{\theta})\hat{p}_{data}(\mathrm{x})}[\nabla_{\boldsymbol{\theta}} \log P(\mathrm{y}|\mathrm{x}, \boldsymbol{\theta})^2] \tag{55}$$

As shown in equation Eq. (22), generative models are able to reuse the gradient $\boldsymbol{g}$ for calculating FIM. However, this reuse poses a challenge when dealing with conditional distributions. This is because Equation Eq. (52) calculates the expected value over the label y, while Equation Eq. (55) calculates the expected value over the conditional distribution of the model. To address this, we introduce an additional approximation by calculating the expected value over the label y in Eq. (56). This is commonly referred to as the **empirical FIM** in the ML community [28], while the statistics community refers to the approximation in Eq. (54) as the empirical FIM, as explained in Eq. (16). To avoid confusion, we will refer to this as the **conditional empirical FIM**.

$$\hat{\boldsymbol{F}}(\boldsymbol{\theta}) \approx \mathbb{E}_{\mathrm{x,y}\sim \hat{p}_{data}}[\nabla_{\boldsymbol{\theta}} \log P(\mathrm{y}|\mathrm{x}, \boldsymbol{\theta})^2] \tag{56}$$

$$\approx \mathbb{E}_{\mathrm{x,y}\sim \hat{p}_{data}}[\nabla_{\boldsymbol{\theta}} \log P(\mathrm{y}|\mathrm{x}, \boldsymbol{\theta})]^2 = \boldsymbol{g}^2 \tag{57}$$

As seen in Eq. (21), to reuse $\boldsymbol{g}$ in Eq. (52), a non-principled approximation is made by taking the square outside the expectation in Eq. (57). Similar to the generative model in Eq. (22), we can obtain the natural gradient with respect to a minibatch $\mathcal{B}$ as follows:

$$\tilde{\nabla} J(\boldsymbol{\theta}) = \hat{\boldsymbol{F}}^{-1} \nabla J(\boldsymbol{\theta}) \approx -\mathbb{E}_{\mathrm{x,y}\in\mathcal{B}}[\nabla_{\boldsymbol{\theta}} \log P(\mathrm{y}|\mathrm{x}, \boldsymbol{\theta})]/\mathbb{E}_{EMA}[\mathbb{E}_{\mathrm{x,y}\in\mathcal{B}}[\nabla_{\boldsymbol{\theta}} \log P(\mathrm{y}|\mathrm{x}, \boldsymbol{\theta})]^2] \tag{58}$$

$$= -\boldsymbol{g}/\mathbb{E}_{EMA}[\boldsymbol{g}^2] \tag{59}$$

Despite the numerous non-trivial approximations made to FIM, it is remarkable that Adam has achieved such great success. Eliminating these non-trivial approximations could be a promising direction for future research.

B.3 Clipping and Epsilon

Although not mentioned in the original Adam paper, it was later discovered that gradient clipping is essential for Adam [41, 44] and is now used as a standard practice. Adafactor [20] incorporates clipping [10] in the algorithm, because the diagonal empirical FIM can have zero components. Therefore, **clipping is applied to the invariant natural gradient**, and then the clipped gradient is used to calculate the momentum, as demonstrated in Algorithm 1. The Adafactor implementation [68] already incorporates this approach, although it is not explicitly mentioned in the original paper [20], as shown in Algorithm 4. Since clipping is already incorporated into the optimizer, global clipping [10] is not necessary.[10] In our experiments, we did not observe any benefit from applying global clipping.

To prevent division by zero before clipping, ϵ is added. The standard value for ϵ in Adam is 1e-8. Adafactor [20] uses 1e-30, but since it is added inside the square root, this translates to 1e-15 on the Adam scale. Wortsman u.a. [55] recommends using a smaller epsilon value of 1e-15 for large-scale models like LLMs due to the empirical observation that the root mean square (RMS) value of gradients tends to decrease as models get larger and training progresses.

To mitigate the need for manual tuning of ϵ across different models, we propose an **adaptive mechanism that adjusts ϵ based on the gradient RMS values**: $\hat{\epsilon} \leftarrow \min(10^{-8}, 0.01\mathrm{RMS}(\boldsymbol{g}_t))$, as shown in Algorithm 1. Our experiments in Appendix C.1.2 demonstrate that this adaptive approach performs comparably to manual tuning, where optimal epsilon values vary across model sizes and domains.[11]

Moreover, in Algorithm 1, epsilon is raised to the power of 2ρ to ensure invariance with respect to the FIM exponent (in Section 3.4.1). To keep the epsilon values in FAdam comparable to those used in Adam, the exponent is multiplied by 2. [12] EAdam [33] and Adafactor [20] modified the algorithm to accumulate ϵ within EMA, but we did not observe any benefits from this approach in our experiments.

B.4 FAdafactor

Adafactor [20] is modified as shown in Algorithm 2, and we refer to this variant as Fisher Adafactor (FAdafactor).

Algorithm 2 Fisher Adafactor (FAdafactor)

1: **given** $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, $\epsilon_2 = 0.01$, $c = 1$, $\lambda = 0.001$, $\rho = 0.5$, η_t
2: **initialize** $\boldsymbol{\theta}_0$, $t \leftarrow 0$, $\boldsymbol{m}_0 \leftarrow 0_{n \times m}$, $\boldsymbol{R}_0 \leftarrow 1_n$, $\boldsymbol{C}_0 \leftarrow 1_m^\top$ ▷ FIM init to 1 as per Section 3.4.4
3: **repeat**
4:    $t \leftarrow t + 1$
5:    $\boldsymbol{g}_t \leftarrow \nabla_{\boldsymbol{\theta}} \log P_t(\boldsymbol{\theta}_{t-1})$ ▷ Stochastic gradient $\boldsymbol{g}_t \in \mathbb{R}^{n \times m}$ as per Eq. (12)
6:    $\hat{\beta}_2 \leftarrow \beta_2(1 - \beta_2^{t-1})/(1 - \beta_2^t)$ ▷ Bias correction as per Section 3.4.4
7:    $\boldsymbol{R}_t \leftarrow \hat{\beta}_2 \boldsymbol{R}_{t-1} + (1 - \hat{\beta}_2)\boldsymbol{g}_t^2 1_m$ ▷ EMA column vector $\boldsymbol{R}_t \in \mathbb{R}^n$
8:    $\boldsymbol{C}_t \leftarrow \hat{\beta}_2 \boldsymbol{C}_{t-1} + (1 - \hat{\beta}_2)1_n^\top \boldsymbol{g}_t^2$ ▷ EMA row vector $\boldsymbol{C}_t \in \mathbb{R}^m$
9:    $\boldsymbol{f}_t \leftarrow \boldsymbol{R}_t \boldsymbol{C}_t / 1_n^\top \boldsymbol{R}_t$ ▷ Diagonal empirical FIM as per Section 3.4.1
10:    $\hat{\epsilon} \leftarrow \min(\epsilon, \epsilon_2 \mathrm{RMS}(\boldsymbol{g}_t))$ ▷ Adaptive epsilon as per Appendix B.3
11:    $\bar{\boldsymbol{g}}_t \leftarrow \boldsymbol{g}_t / (\boldsymbol{f}_t^\rho + \hat{\epsilon}^{2\rho})$ ▷ Invariant natural gradient as per Eq. (27)
12:    $\bar{\boldsymbol{g}}_t \leftarrow \bar{\boldsymbol{g}}_t / \max(1, \mathrm{RMS}(\bar{\boldsymbol{g}}_t)/c)$ ▷ Clip the gradient as per Appendix B.3
13:    $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\bar{\boldsymbol{g}}_t$ ▷ EMA momentum as per Section 3.4.2
14:    $\bar{\boldsymbol{g}}_w \leftarrow \boldsymbol{\theta}_{t-1} / (\boldsymbol{f}_t^\rho + \hat{\epsilon}^{2\rho})$ ▷ Weight decay as per Eq. (28)
15:    $\bar{\boldsymbol{g}}_w \leftarrow \bar{\boldsymbol{g}}_w / \max(1, \mathrm{RMS}(\bar{\boldsymbol{g}}_w)/c)$ ▷ Clip weight decay as per Appendix B.3
16:    $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_t(\boldsymbol{m}_t + \lambda \bar{\boldsymbol{g}}_w)$ ▷ Update $\boldsymbol{\theta}$ as per Eq. (13)
17: **until** *stopping criterion is met*
18: **return** optimized parameters $\boldsymbol{\theta}_t$

---

[10]This feature is often included in ML frameworks as a legacy practice without a clear justification.

[11]We attempted to adjust ϵ based on the weight size using the formula $\hat{\epsilon} = \epsilon/\mathrm{size}(\boldsymbol{\theta})$. However, this approach did not improve performance, likely due to the resulting epsilon values for bias and layer normalization weights becoming unsuitable.

[12]Since the default ρ is 0.5, $\epsilon^{2\rho}$ is typically equal to ϵ.

B.5 Adam and Adafactor

To facilitate comparison between the original algorithms, we present Adam (Algorithm 3) and Adafactor (Algorithm 4), alongside FAdam (Algorithm 1) and FAdafactor (Algorithm 2). Since the full Adafactor algorithm is not explicitly detailed in the original paper [20], we refer the original implementation [68].

Remarkably, Adafactor [20] had already empirically discovered and implemented several of the results we derived from our mathematical framework: applying natural gradient to momentum (Section 3.4.2), omitting bias correction for momentum (Section 3.4.2), and clipping gradients (Appendix B.3). This is akin to the invention of the steam engine before the establishment of thermodynamics, and these empirical findings significantly bolstered our confidence in developing our theory. However, it is peculiar that the momentum calculation incorporates the learning rate, while weight decay remains independent of the learning rate. Accumulating ϵ through EMA is also an unconventional approach.

Algorithm 3 Adam [12]

1: **given** $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, $c = 1$, $\lambda = 0.001$, η_t
2: **initialize** $\boldsymbol{\theta}_0$, $t \leftarrow 0$, $\boldsymbol{m}_0 \leftarrow 0_N$, $\boldsymbol{f}_0 \leftarrow 0_N$
3: **repeat**
4:      $t \leftarrow t + 1$
5:      $\boldsymbol{g}_t \leftarrow \nabla_{\boldsymbol{\theta}} \log P_t(\boldsymbol{\theta}_{t-1})$
6:      $\boldsymbol{f}_t \leftarrow (\beta_2 \boldsymbol{f}_{t-1} + (1 - \beta_2)\boldsymbol{g}_t^2)/(1 - \beta_2^t)$
7:      $\boldsymbol{m}_t \leftarrow (\beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\boldsymbol{g}_t)/(1 - \beta_1^t)$
8:      $\bar{\boldsymbol{g}}_t \leftarrow \boldsymbol{m}_t/(\sqrt{\boldsymbol{f}_t} + \epsilon)$
9:      $\bar{\boldsymbol{g}}_t \leftarrow \bar{\boldsymbol{g}}_t / \max(1, \mathrm{RMS}(\bar{\boldsymbol{g}}_t)/c)$
10:     $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_t(\bar{\boldsymbol{g}}_t + \lambda \boldsymbol{\theta}_{t-1})$
11: **until** *stopping criterion is met*
12: **return** optimized parameters $\boldsymbol{\theta}_t$

Algorithm 4 Adafactor [20, 68]

1: **given** $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-30}$, $c = 1$, $\lambda = 0.001$, η_t
2: **initialize** $\boldsymbol{\theta}_0$, $t \leftarrow 0$, $\boldsymbol{m}_0 \leftarrow 0_{n \times m}$, $\boldsymbol{R}_0 \leftarrow 0_n$, $\boldsymbol{C}_0 \leftarrow 0_m^\top$
3: **repeat**
4:      $t \leftarrow t + 1$
5:      $\boldsymbol{g}_t \leftarrow \nabla_{\boldsymbol{\theta}} \log P_t(\boldsymbol{\theta}_{t-1})$
6:      $\hat{\beta}_2 \leftarrow \beta_2(1 - \beta_2^{t-1})/(1 - \beta_2^t)$
7:      $\boldsymbol{R}_t \leftarrow \hat{\beta}_2 \boldsymbol{R}_{t-1} + (1 - \hat{\beta}_2)(\boldsymbol{g}_t^2 + \epsilon 1_n 1_m^\top)1_m$
8:      $\boldsymbol{C}_t \leftarrow \hat{\beta}_2 \boldsymbol{C}_{t-1} + (1 - \hat{\beta}_2)1_n^\top(\boldsymbol{g}_t^2 + \epsilon 1_n 1_m^\top)$
9:      $\boldsymbol{f}_t \leftarrow \boldsymbol{R}_t \boldsymbol{C}_t / 1_n^\top \boldsymbol{R}_t$
10:     $\bar{\boldsymbol{g}}_t \leftarrow \boldsymbol{g}_t / \sqrt{\boldsymbol{f}_t}$
11:     $\bar{\boldsymbol{g}}_t \leftarrow \bar{\boldsymbol{g}}_t / \max(1, \mathrm{RMS}(\bar{\boldsymbol{g}}_t)/c)$
12:     $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\eta_t \bar{\boldsymbol{g}}_t$
13:     $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - (\boldsymbol{m}_t + \lambda \boldsymbol{\theta}_{t-1})$
14: **until** *stopping criterion is met*
15: **return** optimized parameters $\boldsymbol{\theta}_t$

B.6 Convergence proof

While the original Adam paper [12] presented a convergence proof, Reddi u.a. [30] later demonstrated that this proof was flawed. Subsequently, Défossez u.a. [40] provided a corrected convergence proof for Adam. Our analysis follows the approach of Défossez u.a. [40].

To the best of our knowledge, there is no existing convergence proof for Adam with clipping, despite its widespread use in practice. This is likely due to the non-trivial nature of incorporating clipping into the analysis. Existing convergence proofs only cover SGD with clipping [62]. Therefore, we also present a convergence proof that excludes the effects of clipping.

The convergence proof for FAdam is simpler than that of Adam. Adam's convergence proof needs to consider both the 1st and 2nd momentums simultaneously, as they both influence the gradient update. In contrast, FAdam first computes the natural gradient using FIM and then applies momentum to the natural gradient. This process of applying momentum to the natural gradient is analogous to Polyak momentum applied to SGD. It is well-known that Polyak momentum tightens the convergence bound [23]. Since FAdam's momentum is analogous to Polyak momentum, FAdam's momentum also tightens the convergence bound. Therefore, the convergence bound for the natural gradient without momentum is looser than the convergence bound for FAdam. For the sake of simplicity, we focus on deriving the convergence bound for the natural gradient without momentum. A tighter theoretical convergence bound, incorporating the effects of momentum, is left for future work.

FAdam without the effects of momentum is equivalent to the Adam algorithm with $\beta_1 = 0$. The convergence of Adam under this condition is shown in Eq. (15) of Défossez u.a. [40] as follows:

$$\mathbb{E}\left[||\nabla F(x_\tau)||^2\right] \leq \frac{F(x_0) - F_*}{\alpha_1 \sqrt{N}} + \frac{1}{\sqrt{N}}(4dR^2 + \alpha_1 dRL)\left(\ln\left(1 + \frac{RN}{\epsilon}\right) + \frac{N}{N-1}\right) \quad (60)$$

$$\sim O(d\ln(N)/\sqrt{N}) \quad (61)$$

In Eq. (60), $x \in \mathbb{R}^d$ represents the model parameters $\boldsymbol{\theta}$, α_1 is the learning rate, N represents the number of iterations, and τ_N denotes a random index within the set $\{0, \ldots, N-1\}$. To derive the equation, the following assumptions are made:

The function F is bounded below by F_*, that is:

$$F(x) \geq F_* \text{ for all } x \in \mathbb{R}^d.$$

The l_∞ norm of the stochastic gradients is uniformly almost surely bounded, meaning there exists $R \geq \sqrt{\epsilon}$ such that:

$$||\nabla F(x)||_\infty \leq R - \sqrt{\epsilon} \text{ for all } x \in \mathbb{R}^d.$$

The objective function is smooth, specifically, its gradient is L-Lipschitz continuous with respect to the l_2-norm:

$$||\nabla F(x) - \nabla F(y)||_2 \leq L||x - y||_2 \text{ for all } x, y \in \mathbb{R}^d.$$

Therefore, from Eq. (61), as the number of iterations N increases, the expected value of the squared norm of the gradient diminishes, indicating that F converges to the optimal value F_*.

C   Experiment

C.1   Ablation study

C.1.1   Exponent of FIM

In Section 3.4.1, we provided a theoretical explanation for the use of the square root in FIM, as shown in Eq. (27). To explore alternative exponent values, we conducted experiments, the results of which are presented in Fig. 1d and Table 2. Exponents above 0.5 exhibit relative stability, while values below 0.3 demonstrate a sharp decline in performance. The standard value of 0.5 for the FIM exponent in Adam [12] appears to be a well-justified choice.

Model	Metric	Steps	ρ=0.3	ρ=0.4	ρ=0.5	ρ=0.6	ρ=0.7	ρ=0.8	ρ=0.9	ρ=1.0
LLM 1B	loss	1.5k	5.4	3.9	**3.8**	3.9	4.2	4.3	4.3	4.3
ASR 100M	WER	8k	43.9	11.2	**6.04**	6.08	**6.04**	6.08	6.23	6.33

Table 2: Effects of Varying FIM Exponent on LLM and ASR Model Performance

C.1.2 Epsilon

We conducted experiments to evaluate the effectiveness of an adaptive epsilon strategy (as mentioned in Appendix B.3) and to investigate the impact of different (non-adaptive) epsilon values on the performance of LLM (1B), ASR (600M), and VQ-VAE (100M) models. The results are presented in Table 3.

While the optimal epsilon value varies across different size and domain models, the adaptive epsilon approach demonstrated competitive performance without requiring manual tuning, highlighting its potential for broader applicability. Notably, manual tuning sometimes led to divergence issues with smaller epsilon values like 1e-20 or 1e-30.

Model	Metric	Steps	Adaptive	1e-8	1e-12	1e-15	1e-20	1e-30
LLM 1B	loss	80k	**2.39**	2.47	**2.42**	2.46	NaN	NaN
ASR 600M	WER	8k	**2.04**	2.08	2.07	2.08	**2.05**	2.13
VQ-VAE 100M	FID	50k	**19.5**	20.9	19.9	**19.6**	19.7	NaN

Table 3: Impact of Epsilon on Model Performance (Lower is Better)

C.1.3 Epsilon2

In Algorithm 1, when the root mean square (RMS) of the gradient is very small, $\hat{\epsilon}$ is calculated by multiplying the RMS(grad) by ϵ_2. As shown in Table 3, a ϵ_2 value of 0.01 consistently yielded good performance across all domains, with a ϵ value of 1e-8.

Model	Metric	Steps	1e-1	1e-2	1e-3	1e-4	1e-6	1e-8
LLM 1B	loss	60k	2.63	**2.41**	2.48	2.45	2.60	**2.41**
ASR 600M	WER	40k	1.98	**1.89**	1.92	1.93	1.90	1.91
VQ-VAE 100M	FID	50k	19.9	**19.5**	19.6	**19.5**	19.7	19.8

Table 4: Impact of Epsilon2 on Model Performance (Lower is Better)

C.1.4 Weight decay

As shown in Eq. (28), the most significant algorithmic change in FAdam compared to Adam is the weight decay mechanism. Therefore, we conducted experiments for weight decay.

In the ASR experiments described in Section 4.2, the baseline model utilized Adam, and we found that the weight decay parameter (λ) used in Adam could generally be reused for FAdam. Our enhanced weight decay mechanism played a significant role in improving FAdam's performance, as shown in Table 5.

LibriSpeech WERs	dev	dev-other	test	test-other	avg
Adam λ=0	1.31	2.55	1.34	2.57	1.94
Adam λ=0.001	1.30	2.54	1.33	2.59	1.93
FAdam λ=0	1.30	2.43	1.34	2.63	1.92
FAdam λ=0.001	1.29	2.49	1.34	2.49	**1.89**

Table 5: LibriSpeech WERs of 600M ASRs with Varying Weight Decay Parameter (λ)

In the LLM experiments described in Section 4.1, the baseline model utilized Adafactor, which, as shown in Algorithm 4, decouples weight decay from the learning rate. Consequently, models using Adafactor require retuning the weight decay parameter λ. While 0.001 was found to be optimal for Adafactor, Table 6 demonstrates that 0.1 worked best for FAdafactor.

λ	1	0.1	0.01	0.001
LLM 1B	3.007	**2.875**	2.998	3.004

Table 6: Eval loss of 1B LLMs with Varying Weight Decay Parameter (λ)

C.2 Experiments Compute Resources

The LLM 1B model was trained on 16 TPUv5 [56] devices (80GB HBM) for one day. Each training example consisted of 2k tokens with a global batch size of 256. The ASR 600M model was fine-tuned on 32 TPUv3 devices (8GB HBM) for half a day. Each utterance had an average duration of 12 seconds with a global batch size of 512. The VQ-VAE 100M model was trained on 16 TPUv4 devices (16GB HBM) for one day. Training was performed using images with a resolution of 256x256 and a global batch size of 256.