Exponential weight averaging as damped harmonic motion

Jonathan Patsenker [*1] **Henry Li** [*1] **Yuval Kluger** [1]

Abstract

The exponential moving average (EMA) is a commonly used statistic for providing stable estimates of stochastic quantities in deep learning optimization. Recently, EMA has seen considerable use in generative models, where it is computed with respect to the model weights, and significantly improves the stability of the inference model during and after training. While the practice of weight averaging at the end of training is well-studied and known to improve estimates of local optima, the benefits of EMA over the course of training is less understood. In this paper, we derive an explicit connection between EMA and a damped harmonic system between two particles, where one particle (the EMA weights) is drawn to the other (the model weights) via an idealized zero-length spring. We then leverage this physical analogy to analyze the effectiveness of EMA, and propose an improved training algorithm, which we call BELAY. Finally, we demonstrate theoretically and empirically several advantages enjoyed by BELAY over standard EMA.

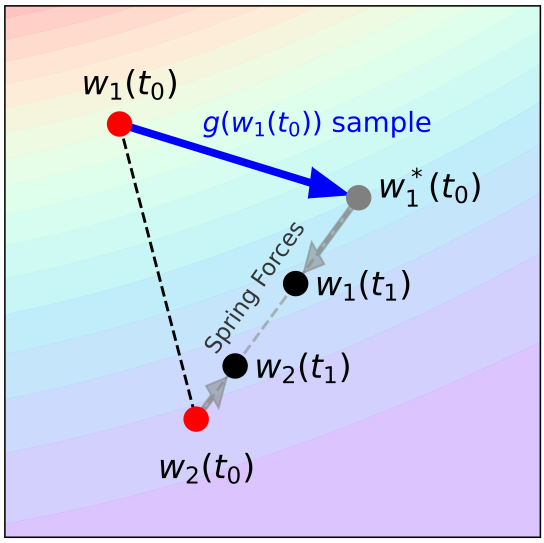

Figure 1. A visualization of the BELAY update step. The background color corresponds to the true full-batch loss function, and g is sampled using an optimizer on a minibatch.

1. Introduction

First-order stochastic gradient optimizers are widely employed in the modern deep learning literature to train large models, and are often motivated by physical analogies such as momentum (Polyak, 1964), curvature (Ypma, 1995), projections (Hestenes et al., 1952; Beck & Teboulle, 2003), and molecular dynamics (Bussi & Parrinello, 2007; Welling & Teh, 2011).

We turn our attention to another common motif in first-order optimization methods, model stabilization via the exponential moving average (EMA). Applied to the gradient of the loss function, EMA is well-understood as a physical approximation of momentum, and this concept is extensively used in many popular adaptive algorithms (Kingma & Ba, 2014; Hinton et al., 2012; Zeiler, 2012; Zhuang et al., 2020) to stabilize gradient descent. We instead focus on EMA applied directly to the model weights. This type of averaging is frequently used in deep learning, especially in generative modeling (Yazıcı et al., 2018; Ho et al., 2019; Song & Ermon, 2020). Weight-based EMA is related to the general idea of weight averaging (Ruppert, 1988; Polyak & Juditsky, 1992), which is known to improve the generalization properties of stochastic algorithms at the end of training (Izmailov et al., 2018). However, a physical analogy for weight averaging has, to our knowledge, not been explored.

In this paper, we establish a clear connection between the weight-based EMA update and the discrete time Euler update of a damped harmonic oscillator. In other words, EMA can be modeled by an idealized zero-length spring that is attached on one end to the model weights, and on the other to the EMA weights. This analogy allows us to highlight several distinct theoretical properties of the weight-based EMA. Finally, we propose BELAY, which explores a variant of EMA where the model weights can also be updated by the EMA weights, and find that BELAY confers increased robustness of the optimization algorithm to the learning rate.

2. Background

For a deep neural network with parameters $\boldsymbol{w} \in \mathbb{R}^n$, we aim to minimize some loss function $\mathcal{L} : \boldsymbol{w} \mapsto \mathbb{R}$. For applications of generative modeling, $\mathcal{L}$ is often negative log-likelihood (Salimans et al., 2017; Dinh et al., 2014) or a score matching loss (Hyvärinen & Dayan, 2005; Vincent, 2011; Song et al., 2020a; Ho et al., 2020). First-order stochastic gradient optimizers iteratively sample an estimate for $\nabla\mathcal{L}(\boldsymbol{w})$ and update $\boldsymbol{w}$ with this information. In this work, we refer to this 'update' for an optimizer as a function $g(\boldsymbol{w}) \in \mathbb{R}^n$.

2.1. The Exponential Moving Average

Using the time-averaged parameters of a model over the course of training is a commonly used technique in generative modeling across GANs (Salimans et al., 2016; Yazıcı et al., 2018), normalizing flows (Ho et al., 2019), autoregressive models (Salimans et al., 2017; Child et al., 2019), and diffusion models (Song & Ermon, 2020; Song et al., 2020b; Ho et al., 2020). The exponential moving average (EMA) of a set of parameters $\boldsymbol{w}$ over time can be described recursively in terms of the time update

$$\begin{aligned} \boldsymbol{w}^{EMA}(t+1) &= \alpha \boldsymbol{w}^*(t) + (1-\alpha)\boldsymbol{w}^{EMA}(t) \\ \boldsymbol{w}^*(t) &= \boldsymbol{w}(t) + g(\boldsymbol{w}(t)) \end{aligned} \tag{1}$$

where $\alpha \in [0, 1]$. Weighted-based EMA has been shown to improve model stability over the course of training (Song & Ermon, 2020; Yazıcı et al., 2018).

2.2. Damped Harmonic Oscillators

Let $\boldsymbol{w}_1, \boldsymbol{w}_2 : [0, T] \to \mathbb{R}^n$ denote the positions of two point particles with masses m_1, m_2, connected by an idealized zero-length spring. In a damped harmonic system, the force exerted by w_2 on w_1 can be written as

$$\boldsymbol{F} = \underbrace{k(\boldsymbol{w}_2 - \boldsymbol{w}_1)}_{\boldsymbol{F}_S} \underbrace{- c_1 \dot{\boldsymbol{w}}_1}_{\boldsymbol{F}_D}, \tag{2}$$

where $\boldsymbol{F}_S$ is the spring force given by Hooke's Law and spring constant k, and $\boldsymbol{F}_D$ is the damping force. External forces on $\boldsymbol{w}_1$ may be modeled with a function $f : \mathbb{R}^n \times [0, T] \to \mathbb{R}^n$. Applying Newton's second law of motion provides the ODE

$$\ddot{\boldsymbol{w}}_1 = \frac{k}{m_1}(\boldsymbol{w}_2 - \boldsymbol{w}_1) - \frac{c_1}{m_1}\dot{\boldsymbol{w}}_1 + \frac{1}{m_1} f(\boldsymbol{w}_1, t). \tag{3}$$

Using the second kinematics equation and discretizing t with time-steps of size Δt, we have

$$\begin{aligned} \boldsymbol{w}_1(t+\Delta t) &= \boldsymbol{w}_1(t) + \dot{\boldsymbol{w}}_1(t)\Delta t + \frac{1}{2}\ddot{\boldsymbol{w}}_1(t)\Delta t^2 \\ &= \boldsymbol{w}_1(t) + \frac{\Delta t^2}{2m_1} f(\boldsymbol{w}_1(t), t) \\ &\quad + \frac{k\Delta t^2}{2m_1}(\boldsymbol{w}_2(t) - \boldsymbol{w}_1(t)) \\ &\quad + \left(\Delta t - \frac{c_1 \Delta t^2}{2m_1}\right)\dot{\boldsymbol{w}}_1(t) \end{aligned} \tag{4}$$

The position of the other mass, $\boldsymbol{w}_2$ can be similarly derived. For the purposes of this work, $\boldsymbol{w}_2$ is not subject to any external force. We thus have

$$\begin{aligned} \boldsymbol{w}_2(t+\Delta t) &= \boldsymbol{w}_2(t) + \frac{k\Delta t^2}{2m_2}(\boldsymbol{w}_1(t) - \boldsymbol{w}_2(t)) \\ &\quad + \left(\Delta t - \frac{c_2 \Delta t^2}{2m_2}\right)\dot{\boldsymbol{w}}_2(t), \end{aligned} \tag{5}$$

where $c_2 \in \mathbb{R}^+$ is a damping constant analogous to c_1. Together these two equations can be used to describe to perform Euler integration and solve an initial value problem when $\boldsymbol{w}_1(0), \boldsymbol{w}_2(0)$ are known.

3. EMA as Damped Harmonic Motion

We relate Eqs. 4 and 5 to weight-based EMA, described in Eq. 1, by choosing the damping parameter $c_1 = \frac{2m_1}{\Delta t}$, and setting $\beta = 1 - \frac{k\Delta t^2}{2m_1}$. Thus, we can rewrite Equation 4 as

$$\begin{aligned} \boldsymbol{w}_1(t+\Delta t) &= \boldsymbol{w}_1(t) + \beta g'(\boldsymbol{w}_1(t)) + (1-\beta)(\boldsymbol{w}_2(t) - \boldsymbol{w}_1(t)) \\ &= \beta \boldsymbol{w}_1(t) + \beta g'(\boldsymbol{w}_1(t)) + (1-\beta)\boldsymbol{w}_2(t) \\ &= \beta \boldsymbol{w}_1^*(t) + (1-\beta)\boldsymbol{w}_2(t), \end{aligned} \tag{6}$$

where $g' = \frac{\Delta t^2}{2\beta m_1} f(\cdot, t)$, and $\boldsymbol{w}_1^*(t) = \boldsymbol{w}_1(t) + g'(\boldsymbol{w}_1(t))$.

Selecting $c_2 = \frac{2m_2}{\Delta t}$, and $\alpha = 1 - \frac{k\Delta t^2}{2m_2}$, we can also rewrite Equation 5 as

$$\boldsymbol{w}_2(t+\Delta t) = \alpha \boldsymbol{w}_1(t) + (1-\alpha)\boldsymbol{w}_2(t). \tag{7}$$

Taking the point-masses $\boldsymbol{w}_1$, $\boldsymbol{w}_2$ to be the model weights $\boldsymbol{w}$ and $\boldsymbol{w}^{EMA}$ respectively, we obtain a weight averaging method that is precisely EMA when $\beta = 1$. This occurs when $m_1 \to \infty$. In this case, $\beta \to 1$ and Eq. 6 becomes $\boldsymbol{w}_1(t + \Delta t) = \boldsymbol{w}^*$, which follows line 2 of Eq. 1 (and therefore Eq. 5 follows line 1 of Eq. 1). This implies that EMA has a physical interpretation as a damped harmonic oscillator.

According to our physical interpretation of the standard EMA scheme, $\boldsymbol{w}^*$ is not affected by $\boldsymbol{w}^{EMA}$ during training

since it has infinite mass. In the next section, we explore the possibility of finite m_1. This allows us to harness the strong model stability properties of EMA to guide parameter updates during training, which we evaluate in Section 4.

BELAY We introduce BELAY: (B)ridging (E)xponentia(L) moving (A)verages with spring s(Y)stems, a novel class of optimization methods that generalizes weight averaging by re-expressing each step as a forward Euler integration update to a damped harmonic oscillator. BELAY is parameterized by (k, m_1, m_2, c_1, c_2), and weight update function $g(\boldsymbol{w}, t)$ provided by some existing optimizer. We add the parameter $\eta \in \mathbb{R}^+$ to designate learning rate for the optimizer. The physical interpretation of each parameter is described in Table B.1. We formalize the BELAY algorithm in Algorithm 1, and visualize one update step in Figure 1. We note that when c_1, c_2, are set as described above, the implementation can be simplified to avoid storing or computing momentum terms $\dot{\boldsymbol{w}}(t)$. Even with a non-trivial set of damping parameters, the time and memory overhead of BELAY is negligible compared to conventional training, as is the case in (Izmailov et al., 2018).

Algorithm 1 BELAY

Input: Parameters (k, m_1, m_2, c_1, c_2), weight update function g, learning rate η.
Initialize $t = 0$
Initialize $\boldsymbol{w}_1(0)$, $\boldsymbol{w}_2(0)$ with small random values
Initialize $\dot{\boldsymbol{w}}_1(0) = 0$, $\dot{\boldsymbol{w}}_2(0) = 0$
while stopping criterion not met **do**
  Compute weight-update $g(\boldsymbol{w}_1(t), t)$ {see Table B.1 for details}
  Compute optimizer step $\boldsymbol{w}_1^* = \boldsymbol{w}_1 + \eta g(\boldsymbol{w}_1(t), t)$
  Compute momentum $\boldsymbol{M}_1 = (1 - \frac{c_1}{2m_1})\dot{\boldsymbol{w}}_1(t)$
  Compute momentum $\boldsymbol{M}_2 = (1 - \frac{c_2}{2m_2})\dot{\boldsymbol{w}}_2(t)$
  $\alpha = 1 - \frac{k}{2m_1}$
  $\beta = 1 - \frac{k}{2m_2}$
  Update $\boldsymbol{w}_1(t+1) = \alpha \boldsymbol{w}_1^*(t) + (1-\alpha)\boldsymbol{w}_2(t) + \boldsymbol{M}_1$
  Update $\boldsymbol{w}_2(t+1) = \beta \boldsymbol{w}_2(t) + (1-\beta)\boldsymbol{w}_1(t) + \boldsymbol{M}_2$
  $\boldsymbol{\delta v}_1 = \frac{k}{m_1}(\boldsymbol{w}_2 - \boldsymbol{w}_1) - \frac{c_1}{m_1}\dot{\boldsymbol{w}}_1(t) + \frac{\eta}{2\alpha} g(\boldsymbol{w}_1(t), t)$ [1]
  $\boldsymbol{\delta v}_2 = \frac{k}{m_2}(\boldsymbol{w}_1 - \boldsymbol{w}_2) - \frac{c_2}{m_2}\dot{\boldsymbol{w}}_2(t)$
  Update $\dot{\boldsymbol{w}}_1(t+1) = \dot{\boldsymbol{w}}_1 + \boldsymbol{\delta v}_1$
  Update $\dot{\boldsymbol{w}}_2(t+1) = \dot{\boldsymbol{w}}_2 + \boldsymbol{\delta v}_2$
  $t = t + 1$
end while

[1] We scale g by $\frac{\eta}{2\alpha}$ because we earlier re-expressed $\frac{1}{2\alpha m_1} g$ as ηg to simplify the update to $\boldsymbol{w}_1(t)$ (since $\Delta t = 1$). Since $\boldsymbol{\delta v}_1 = \ddot{\boldsymbol{w}}_1(t) = \frac{k}{m_1}(\boldsymbol{w}_2 - \boldsymbol{w}_1) - \frac{c_1}{m_1}\dot{\boldsymbol{w}}_1(t) + \frac{1}{m_1} g(\boldsymbol{w}_1(t), t)$, and we would like to retain scaling by η as a parameter, we can use $\frac{\eta}{2\alpha}$.

3.1. Connection to Optimizers with Momentum

In BELAY, the damping coefficients c_1 and c_2 explicitly control the momentum term in every weight update step. Even when these coefficients are set to cancel the momentum (as described in Section 3) however, BELAY still uses momentum information. When running SGD with momentum, the weights are updated with the step,

$$\begin{aligned} \boldsymbol{v}(t) &= \lambda \nabla \mathcal{L}(\boldsymbol{w}(t)) + (1-\lambda)\boldsymbol{v}(t-1) \\ \boldsymbol{w}(t+1) &= \boldsymbol{w}(t) + \alpha \boldsymbol{v}(t) \\ &= \boldsymbol{w}(t) + \alpha \sum_{s=0}^{t} (1-\lambda)^s \lambda \nabla \mathcal{L}(\boldsymbol{w}(t-s)) \end{aligned} \tag{8}$$

where $\mathcal{L}$ is the loss function being optimized and $\alpha \in [0, 1]$. In the case where $\mathcal{L}$ is linear,

$$\begin{aligned} \sum_{s=0}^{t} a_s \nabla \mathcal{L}(\boldsymbol{w}(t-s)) &= \nabla \mathcal{L}\left(\sum_{s=0}^{t} a_s \boldsymbol{w}(t-s) \right) \\ &= \nabla \mathcal{L}\left(\boldsymbol{w}^{EMA}(t) \right), \end{aligned} \tag{9}$$

where $a_s = (1-\lambda)^s \lambda$. Given specific parameter choices, this is an update for BELAY. We have shown that the linearization of a BELAY update $\nabla \mathcal{L}(\boldsymbol{w}^{EMA}(t))$ is therefore equivalent to the momentum update term $\boldsymbol{v}(t)$. This relationship is analogous to that of SWA and FGE documented by (Izmailov et al., 2018).

3.2. Deriving the Spring Constant k

One caveat of vanilla EMA is its sensitivity to α, which governs the exponential decay of the moving average. Picking α too large causes the average to converge too slowly, which can drastically extend the training time of a learning algorithm. Conversely, too small α will overly favor the present iterate, which at best increases model variance at evaluation-time, and at worst defeats the intended purpose of the moving average. The challenge of choosing a proper α is further complicated by its dependence on the total runtime of the training protocol, in terms of the number of gradient steps T.

We leverage our physical analogy to harmonic oscillators to choose k such that the system is invariant to T. We achieve this by examining the closed form solution of an overdamped spring system:

$$\boldsymbol{w}(t) = C_1 e^{\left(-\delta + \sqrt{\delta^2 - \frac{k}{m}}\right)t} + C_2 e^{\left(-\delta - \sqrt{\delta^2 - \frac{k}{m}}\right)t} \tag{10}$$

where $\delta = \frac{1}{\Delta t}$. Using the initial conditions $\dot{\boldsymbol{w}}(0) = 0$ and $\boldsymbol{w}(0) = \mathbf{x}_0$, we may obtain integration constants C_1 and C_2, and then solving for the function $k(T)$ such that $\boldsymbol{w}(T) = \boldsymbol{w}(T_0)$, we obtain $k \approx k_0 \frac{T_0}{T}$. In our experiments, we let $k_0 = 1$ and $T_0 = 1e6$. From this, we obtain our proposed T-invariant spring constant $k(T) = \frac{C}{T}$, where $C = 1e6$.

4. Numerical Experiments

In this section, we evaluate BELAY on a set of synthetic and real optimization tasks, and demonstrate several empirical properties of the convergence behavior of BELAY.

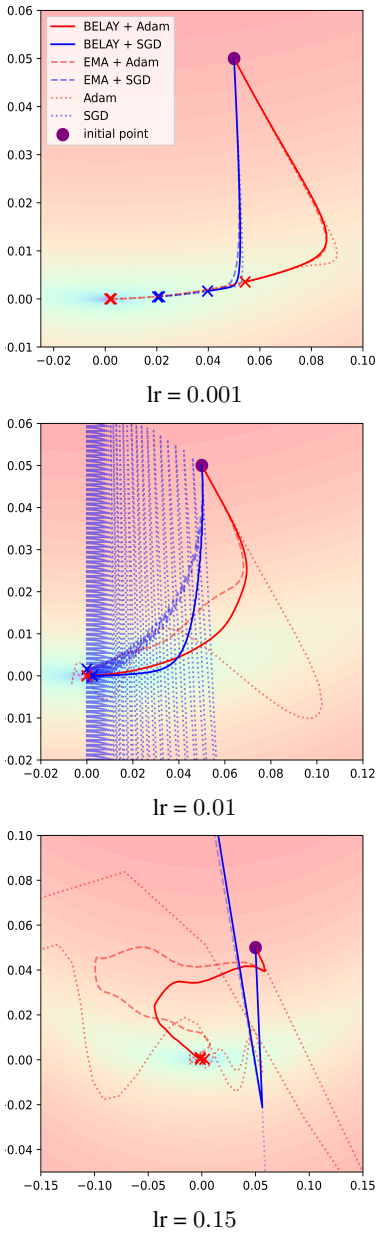

Figure 2. Comparisons between BELAY or EMA + Adam, BELAY or EMA + SGD, Adam and SGD at different learning rates run on the Rosenbrock function. Parameter and function details are located in Appendix C.1.

Robustness to Learning Rate First, we compare the performance of BELAY to EMA, Adam, and SGD on a set of ill-conditioned 2D examples. Figure 2 shows how the stability of each optimizer varies w.r.t. the learning rate. As expected, we find that Adam is generally much more stable than SGD across different learning rates. In the high-learning rate regime, only BELAY and EMA, applied to Adam, are capable of converging without serious instability. In the medium learning rate regime, we see that weight averaging also reduces the tendency of the SGD update to diverge. Overall, weight averaging is better suited to stabilizing the trajectory of an optimizer than momentum alone, especially with higher learning rates.

This analysis is highly applicable to training deep neural networks due to the inherent link between learning rate, and loss function smoothness (Bottou et al., 2018; Wu et al., 2018). Because the latter quantity may vary across weight values, it is important for an optimizer to be robust to learning rate, so that it is able to provide a stable minimizing trajectory across different levels of smoothness without sacrificing speed of convergence.

Faster Convergence We further analyze speed of convergence of the above algorithms in Figure 3. We find that BELAY with either Adam or SGD is able to consistently be among the fastest algorithms to converge. Note that in certain examples, such as the Beale function, momentum-based methods may struggle due to the existence of a nearby saddle-point (off to the left of the field of vision), however weight-averaging is able to remedy this.

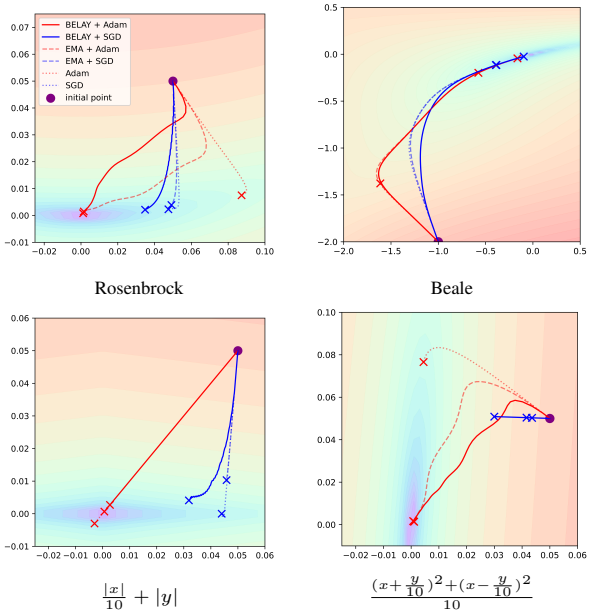

Figure 3. Comparison of trajectories and speed of convergence for BELAY, EMA, and no weight averaging with both Adam and SGD. Each run is stopped near where BELAY has converged. Parameter and function details are located in Appendix C.2

Generative Modeling Finally, we compare BELAY against EMA on a generative modeling task, on two datasets: CIFAR10 and MNIST. More experiments with generative modelling are located in Appendix C.4. We see that BELAY compares favorably with respect to competing algorithms.

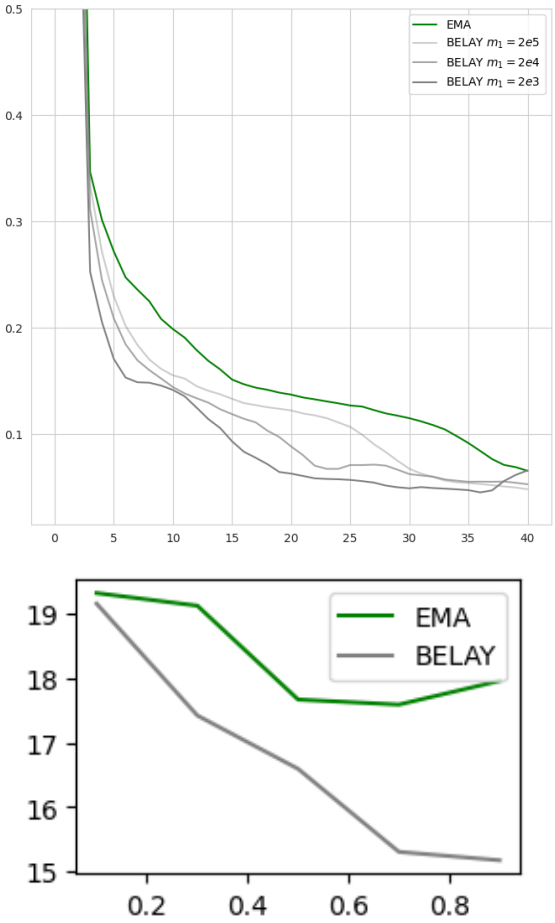

Figure 4. Comparison between loss curves (top) and FID score across training iterations (bottom) of BELAY and EMA, trained to generate MNIST. Details in Appendix C.3

Method	Train Loss	Test Loss	FID
BELAY	0.042	0.040	15.2
EMA	0.056	0.061	18.1

Table 4.1. Comparison of final losses and FID scores of BELAY and EMA, run to generate MNIST digits. Details in Appendix C.3.

5. Conclusion and Future Work

In this work, we have presented BELAY: a simple to implement and efficient weight-averaging method that uses physical systems to bridge the understanding between EMA and momentum based methods. We have shown how BELAY adds a new set of parameters to EMA, each with a strong and intuitive physical meaning, and in turn have shown how EMA uses momentum information, even when run on a non-momentum based method. Furthermore, we have shown empirically, that BELAY may outperform EMA in certain cases, implying that the choice of parameters is non-trivial. We have suggested some ways to set parameters to achieve desired behaviours.

With the basis outlined in this work, many future directions are apparent. First, by leveraging our newfound physical intuition, new theoretical guarantees for momentum-based methods as well as weight averaging could be drawn, by proving convergence bounds on BELAY. A deeper dive into the damping parameter, and the interplay between the spring-based momentum term along with the classical heavy-ball momentum term of the loss function can be investigated to come up with what could be a more guaranteed "critically-damp" regime. Furthermore, large-scale empirical analysis could be performed to select optimal parameters for BELAY, and determine strong heuristics for practitioners interested in large-scale learning tasks.

References

Beck, A. and Teboulle, M. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Bottou, L., Curtis, F. E., and Nocedal, J. Optimization methods for large-scale machine learning. *SIAM review*, 60(2):223–311, 2018.

Bussi, G. and Parrinello, M. Accurate sampling using langevin dynamics. *Physical Review E*, 75(5):056707, 2007.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Dinh, L., Krueger, D., and Bengio, Y. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Hestenes, M. R., Stiefel, E., et al. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49(6):409–436, 1952.

Hinton, G., Srivastava, N., and Swersky, K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2, 2012.

Ho, J., Chen, X., Srinivas, A., Duan, Y., and Abbeel, P. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, pp. 2722–2730. PMLR, 2019.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Hyvärinen, A. and Dayan, P. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. G. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. *arXiv preprint arXiv:2206.00364*, 2022.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Polyak, B. T. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964.

Polyak, B. T. and Juditsky, A. B. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

Ruppert, D. Efficient estimations from a slowly convergent robbins-monro process. Technical report, Cornell University Operations Research and Industrial Engineering, 1988.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

Song, Y. and Ermon, S. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448, 2020.

Song, Y., Garg, S., Shi, J., and Ermon, S. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pp. 574–584. PMLR, 2020a.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Vincent, P. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011.

Wu, X., Ward, R., and Bottou, L. Wngrad: Learn the learning rate in gradient descent. *arXiv preprint arXiv:1803.02865*, 2018.

Yazıcı, Y., Foo, C.-S., Winkler, S., Yap, K.-H., Piliouras, G., and Chandrasekhar, V. The unusual effectiveness of averaging in gan training. *arXiv preprint arXiv:1806.04498*, 2018.

Ypma, T. J. Historical development of the newton–raphson method. *SIAM review*, 37(4):531–551, 1995.

Zeiler, M. D. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Zhuang, J., Tang, T., Ding, Y., Tatikonda, S. C., Dvornek, N., Papademetris, X., and Duncan, J. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *Advances in neural information processing systems*, 33: 18795–18806, 2020.

A. Derivations

A.1. Derivation of k

Letting $\boldsymbol{w}(0) = \boldsymbol{w}_0$ and $\dot{\boldsymbol{w}}_0 = 0$, we see that Eq. 10 reduces to

$$\boldsymbol{w}_0 = C_1 + C_2 \tag{11}$$

$$0 = C_1\left(-\delta + \sqrt{\delta^2 - \frac{k}{m}}\right) + C_2\left(-\delta - \sqrt{\delta^2 - \frac{k}{m}}\right). \tag{12}$$

Substituting Eq. 11 into Eq. 12, we obtain

$$C_1 = \boldsymbol{w}_0\left(\frac{\delta}{\sqrt{\delta^2 - \frac{k}{m}}} - \frac{1}{2}\right)$$

$$C_2 = \boldsymbol{w}_0\left(\frac{1}{2} - \frac{\delta}{\sqrt{\delta^2 - \frac{k}{m}}}\right).$$

Now we see that the general solution of the harmonic oscillator takes the form

$$\boldsymbol{w}(t) = \boldsymbol{w}_0\left(\frac{\delta}{\sqrt{\delta^2 - \frac{k}{m}}} - \frac{1}{2}\right) e^{\left(-\delta + \sqrt{\delta^2 - \frac{k}{m}}\right)t} + \boldsymbol{w}_0\left(\frac{1}{2} - \frac{\delta}{\sqrt{\delta^2 - \frac{k}{m}}}\right) e^{\left(-\delta - \sqrt{\delta^2 - \frac{k}{m}}\right)t}.$$

Since all terms in our method are functions of Δt, we may let $\delta = \frac{1}{\Delta t} = 1$ without loss in generality. Letting $t = T$, we have

$$\boldsymbol{w}(T) = \boldsymbol{w}_0\left(\frac{1}{\sqrt{1 - \frac{k}{m}}} - \frac{1}{2}\right) e^{\left(-1 + \sqrt{1 - \frac{k}{m}}\right)T} + \boldsymbol{w}_0\left(\frac{1}{2} - \frac{1}{\sqrt{1 - \frac{k}{m}}}\right) e^{\left(-1 - \sqrt{1 - \frac{k}{m}}\right)T}.$$

We would like to choose $k = k(T)$ such that $\boldsymbol{w}(T) = \boldsymbol{w}(T_0)$ for some reference time T_0. We have found $T_0 = 1,000,000$ to be a reasonable default parameter. We see that we approximately satisfy this condition when $k = \frac{k_0}{T}$.

B. Parameters

We describe all the parameters of BELAY in detail with intuitive physical explanations, and appropriate units assigned in Table B.1. We note that, outside of the optimizer $g(\boldsymbol{w}, t)$ and learning rate η, only m_2 is left as a free parameter in EMA. In BELAY, we additionally have $m_1 < \infty$ as a tunable parameter.

Param.	Description	SI Units
k	spring constant, scales the force bringing both masses together. This force can also be controlled by changing both m_1 and m_2 simultaneously.	N/m
m_1	mass of first point-mass, scales force on point-mass inversely. Implicitly built into g', and scales momentum term if c_1 is not chosen to remove momentum.	g
m_2	mass of second point-mass. Analogous to m_1 but on $\boldsymbol{w}_2$, but does not have an effect on g'	g
c_1, c_2	damping coefficients for both masses. Can be set to $\frac{2m}{\Delta t}$ in order to fully dampen the system and ensure a return to 0 velocity between time-steps. Deviating will retain velocity and add a momentum term in an optimization sense.	Ns/m
$g(\boldsymbol{w}, t)$	weight update function, interpretable as a force applied to $\boldsymbol{w}_1$. This value is provided by an existing optimization algorithm. For example for full-batch gradient descent the value is $(-\nabla f)$ for some loss function f.	N
η	scale of weight update, learning rate for gradient descent based methods.	s^2/m

Table B.1. Physical parameters of BELAY and interpretations

C. Implementation Details

C.1. Learning rate robustness experiment

We state all the parameter choices for all runs displayed in Figure 2 here. Since we separately observe in Figure 2 that different methods are robust to different learning rates, we run each optimizer at the highest learning rate that optimizer can handle while producing a stable converging trajectory across all runs.

1. BELAY + Adam is run with the parameter set, $k = 1$, $m_1 = 10$, $m_2 = 20$. Damping coeff. c_1, c_2 are set to zero out the velocity term as stated in Section 3

2. BELAY + SGD is run with the parameter set, $k = 1$, $m_1 = 10$, $m_2 = 20$. Damping coeff. c_1, c_2 are set to zero out the velocity term as stated in Section 3
3. EMA + Adam is run with the parameter $\alpha = 0.95$.
4. EMA + SGD is run with the parameter $\alpha = 0.95$.
5. Adam is run with default parameters.

The Rosenbrock function was chosen with parameters $a = 0$, $b = 100$ for easy visualization:

$$f(x, y) = (a - x)^2 + b(x - (y^2))^2.$$

The following version of the Beale function was used, centering the global minima at (0,0) for easy visualization:

$$\begin{aligned} f(x, y) = &(1.5 - (x + 3) + (x + 3) * (y + 0.5))^2 \\ &+ (2.25 - (x + 3) + (x + 3) * (y + 0.5)^2)^2 \\ &+ (2.625 - (x + 3) + (x + 3) * (y + 0.5)^3)^2 \end{aligned}$$

C.2. Convergence trajectory experiments

We state all the learning rate choices for all runs displayed in Figure 3 here. The values for all parameters except for learning rate were the same as those described in Appendix C.1.

1. BELAY + Adam is run with learning rate $\eta = 5 * 10^{-2}$.
2. BELAY + SGD is run with $\eta = 10^{-2}$.
3. EMA + Adam is run with $\eta = 10^{-2}$.
4. EMA + SGD is run with $\eta = 10^{-3}$.
5. Adam is run with $\eta = 10^{-3}$
6. SGD is run with $\eta = 10^{-3}$

C.3. MNIST Experiment

BELAY is run with parameters $k = 1$, $m_1 = 2000$, $m_2 = 500$, and damping c_1, c_2, to zero out the velocity term as described in Section 3. The Adam optimizer is used with learning rate $\eta = 5 * 10^{-2}$. Both methods are run to optimize the score-based diffusion model described in (Song & Ermon, 2020) on MNIST digits.

C.4. CIFAR Experiment

BELAY is run with parameters ($k = 1, m_1 = 20000, m_2 = 2000$), and damping c_1, c_2, to zero out the velocity term as described in Section 3. Both methods are run to optimize the score-based diffusion model described in (Karras et al., 2022) on CIFAR-10 images.

Method	Train Loss	Test Loss	FID
BELAY	0.176	0.170	1.99
EMA	0.175	0.168	1.98

Table C.1. Comparison of final losses and FID scores of BELAY and EMA, run to generate CIFAR-10 images.

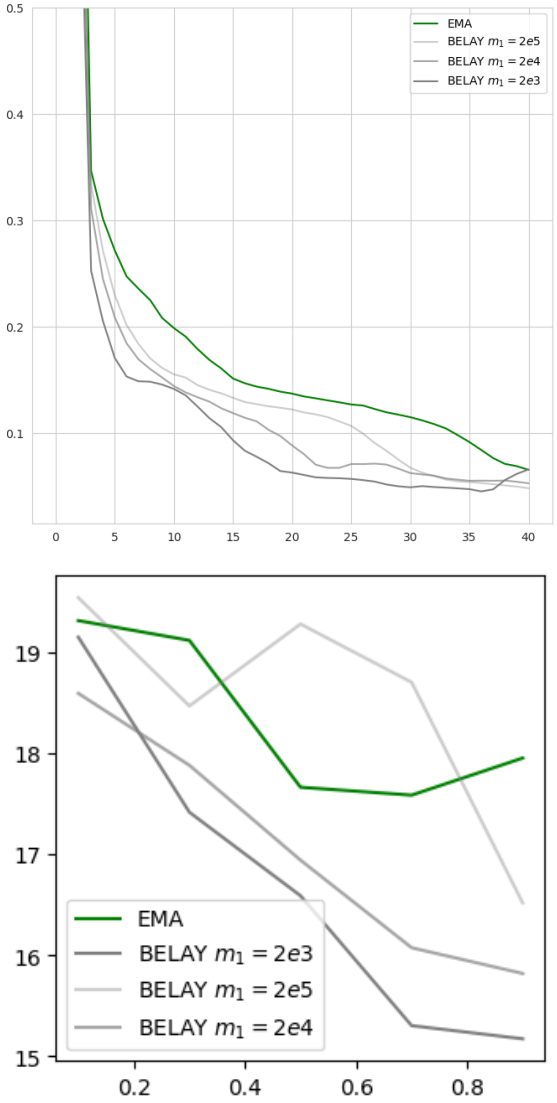

Figure 5. Comparison of train (top) and test loss (bottom) across training iteration of BELAY at parameters $m_1 = 2000, 2 * 10^4, 2 * 10^5$ and EMA, trained to generate MNIST with the score-based diffusion model from (Song & Ermon, 2020)

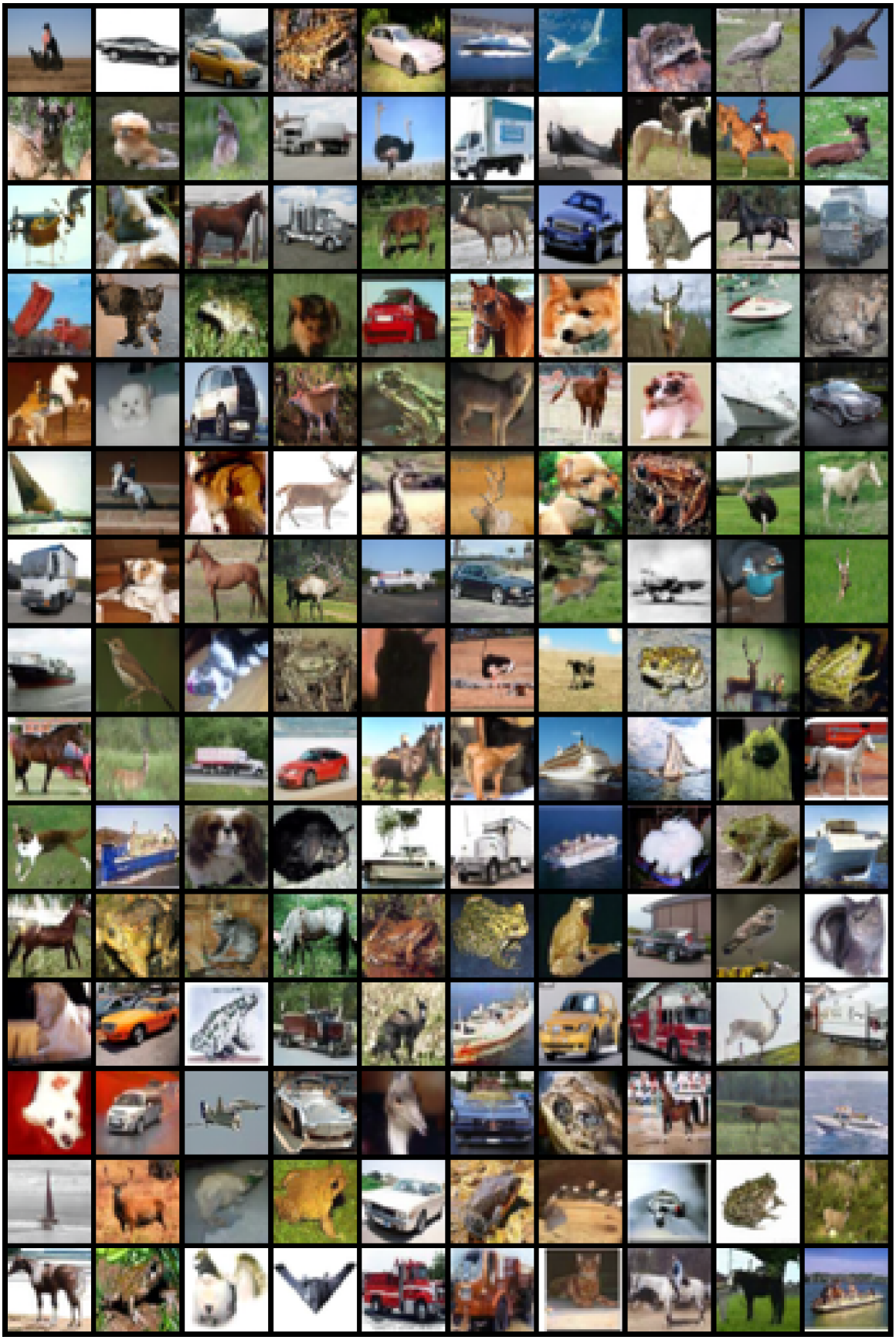

Figure 6. Images from an unconditional diffusion model with EMA trained on the CIFAR10 dataset.

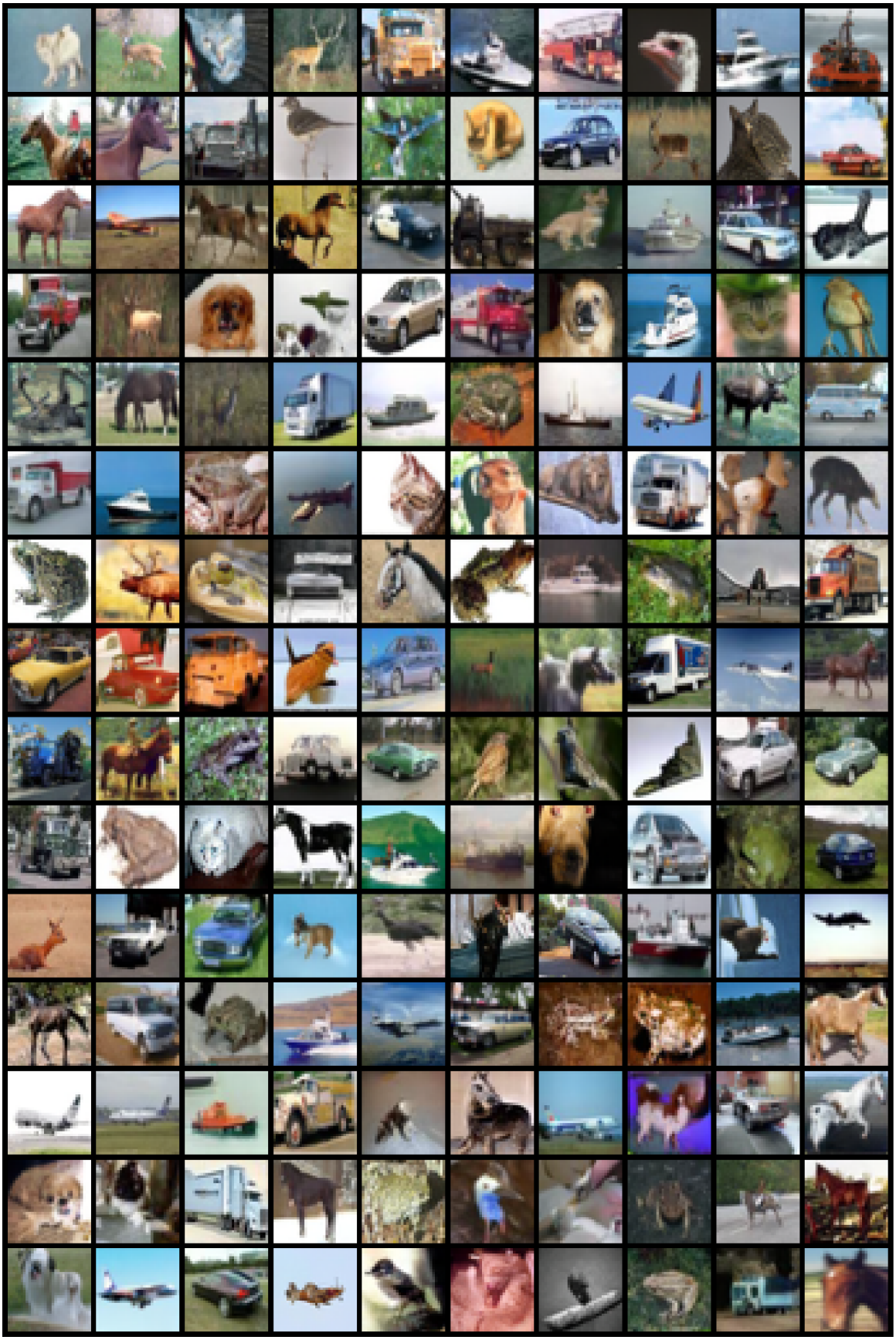

Figure 7. Images from an unconditional diffusion model with BELAY trained on the CIFAR10 dataset.