

CS 182 / EECS 282A Official Course Notes

Fall 2026

Note 4: Momentum, Adam, and AdamW

Based on lectures by Prof. Gireeja Ranade and Prof. Anant Sahai

Prerequisites

Stochastic gradients, basic linear algebra (eigenvalues, unit circle stability), and the momentum recursion covered in Section 4.1.

Reading and coverage map

Prince correspondence (partial: Section 6.4 together with the optimization and regularization notes in Chapters 6 and 9). Prince covers adaptive optimization and weight decay.

JRS prerequisite review (pp. 26–30 and p. 145). These pages review convex optimization, conditioning, gradients, and AdamW/weight-decay ingredients used below.

Beyond Prince. This note develops the momentum eigenmode analysis, the careful separation between SignSGD and Adam, and AdamW decoupling.

4.1 Momentum as a low-pass filter

Poor conditioning can make optimization challenging. A learning rate small enough to remain stable in a direction with a large singular value may make painfully slow progress in a direction with a small singular value. Increasing the learning rate makes the high singular value coordinate overshoot and oscillate. Averaging recent gradients can suppress this high-frequency bouncing, as the one-state RC analogy in Figure 4.1 suggests.

One convention maintains an exponentially weighted gradient state

$$z_{t+1} = \beta z_t + (1 - \beta) \nabla f(w_t), \quad z_0 = 0, \quad 0 \leq \beta < 1, \quad (4.1)$$

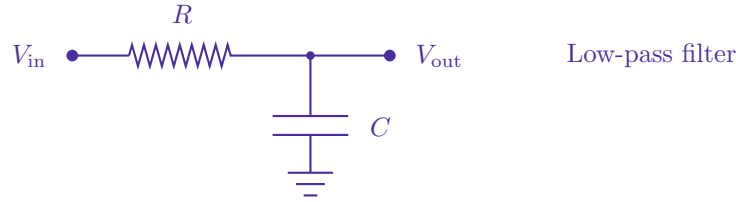
$$w_{t+1} = w_t - \eta z_{t+1}. \quad (4.2)$$

Unrolling Equation (4.1) gives

$$z_{t+1} = (1 - \beta) \sum_{k=0}^t \beta^{t-k} \nabla f(w_k). \quad (4.3)$$

Thus one extra vector stores an entire exponentially decaying history. Plain SGD stores the parameters; this momentum form adds one optimizer-state vector; Adam adds two.

A circuit / physics intuition:



If $V_{\text{in}} = \text{step of mag } V$

$$V_{\text{out}} = V (1 - e^{-t/RC})$$

Figure 4.1: Circuit intuition for momentum. An RC low-pass filter stores one state—capacitor charge. Exponential momentum is the analogous discrete one-state filter for gradients.

Remark

Why not average every past gradient uniformly? The normalized influence of any fixed early gradient then decays only algebraically, like $1/t$, so unusually large stale terms can matter for many steps after the iterate enters a different regime. A short sliding window forgets them abruptly; exponential weighting forgets them geometrically with one state vector rather than storing the window.

Implementation note: two momentum conventions. The recurrence above maintains a *normalized* running average z_t whose $(1 - \beta)$ prefactor ensures z_t has the same scale as any single gradient. Many libraries instead accumulate an *unnormalized* velocity v_t that omits the $(1 - \beta)$ factor. The two conventions and their corresponding weight updates are

$$\begin{aligned} \textbf{Normalized:} \quad z_{t+1} &= \beta z_t + (1 - \beta) \nabla f(w_t), & w_{t+1} &= w_t - \eta_z z_{t+1}, \\ \textbf{Unnormalized:} \quad v_{t+1} &= \beta v_t + \nabla f(w_t), & w_{t+1} &= w_t - \alpha_v v_{t+1}. \end{aligned} \tag{4.4}$$

The only algebraic difference is the $(1 - \beta)$ multiplying the gradient in the first line. Dividing the normalized recurrence through by $(1 - \beta)$ shows that $v_t = z_t / (1 - \beta)$, so the unnormalized velocity is larger by exactly $1/(1 - \beta)$. For a constant gradient $\nabla f(w_t) = c$, the steady states confirm this:

$$z_\infty = c, \quad v_\infty = \frac{c}{1 - \beta}.$$

Because v_t is inflated by the factor $1/(1 - \beta)$, its learning rate must shrink by the same factor to produce identical parameter updates:

$$\alpha_v = \eta_z (1 - \beta). \tag{4.5}$$

For example, with $\beta = 0.9$, a normalized learning rate of $\eta_z = 0.1$ corresponds to an unnormalized rate of $\alpha_v = 0.01$ —an order of magnitude smaller. Nominal learning rates from the two conventions are therefore not directly comparable; the upper panel of Figure 4.2 compares their steady gains after matching the effective parameter step.

Example: The normalized recurrence has unit gain on a persistent gradient. If $\nabla f(w_t) = (-1)^t c$ alternates sign, its steady response has magnitude

$$\frac{\|z_{t+1}\|}{\|c\|} = \frac{1 - \beta}{1 + \beta}. \tag{4.6}$$

The trial form $z_t = A(-1)^t$ verifies the ratio. Momentum therefore attenuates an alternating high-curvature bounce while retaining a persistent direction.

Let us consider a scalar system For a quadratic loss $f(\vec{w}) = \frac{1}{2}\vec{w}^\top H \vec{w}$, rotating into the eigenbasis of the Hessian H decouples the system: each eigendirection with eigenvalue λ_i evolves independently, governed by the scalar loss $f(w) = \frac{1}{2}\lambda_i w^2$. (This is the same SVD decoupling used in the implicit-regularization and SGD-convergence analyses.) Understanding one scalar mode therefore tells us everything about the quadratic case—and gives the right intuition for the nonquadratic setting where eigendirections change at every step.

The two-state recurrence for scalar system For a scalar function $f(w) = \frac{1}{2}\lambda w^2$ with $\lambda > 0$, the gradient is $\nabla f(w) = \lambda w$. Substituting into the normalized-EMA momentum update (Equations (4.1)–(4.2)) couples the parameter w_t and the momentum state z_t into a linear system (Qian 1999):

$$\begin{bmatrix} w_{t+1} \\ z_{t+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 - \eta(1 - \beta)\lambda & -\eta\beta \\ (1 - \beta)\lambda & \beta \end{bmatrix}}_{A_\lambda} \begin{bmatrix} w_t \\ z_t \end{bmatrix}. \quad (4.7)$$

Key question: when does the mode converge, and how fast? Both w_t and z_t are linear combinations of q_+^t and q_-^t , where $q_\pm$ are the eigenvalues of A_λ . The mode converges to zero if and only if both eigenvalues satisfy $|q_\pm| < 1$ (“inside the unit circle”). The convergence rate is governed by the slower-decaying root: $\max(|q_+|, |q_-|)$.

Eliminating z_t gives the equivalent second-order scalar recurrence

$$w_{t+1} = a_\lambda w_t - \beta w_{t-1}, \quad a_\lambda = 1 + \beta - \eta(1 - \beta)\lambda, \quad (4.8)$$

whose characteristic polynomial is $p(q) = q^2 - a_\lambda q + \beta$. The roots are

$$q_\pm = \frac{a_\lambda \pm \sqrt{a_\lambda^2 - 4\beta}}{2}. \quad (4.9)$$

Solutions are $c_+ q_+^t + c_- q_-^t$ for distinct roots and $(c_1 + c_2 t)q^t$ at a repeated root. Thus the mode decays when both roots lie inside the unit circle; the factor t remains harmless for $|q| < 1$.

Stability condition. For a real quadratic $p(q) = q^2 - a_\lambda q + \beta$, the *Schur stability criterion* gives a clean test: both roots lie inside the unit circle if and only if

$$p(1) > 0, \quad p(-1) > 0, \quad \text{and} \quad |\beta| < 1.$$

(The idea: $p(1) > 0$ prevents a root at $+1$; $p(-1) > 0$ prevents a root at -1 ; the product-of-roots condition $|q_+ q_-| = |\beta| < 1$ prevents both roots from escaping outward simultaneously.) Evaluating:

$$p(1) = \eta(1 - \beta)\lambda, \quad p(-1) = 2(1 + \beta) - \eta(1 - \beta)\lambda, \quad (4.10)$$

so the exact stability condition is

$$\boxed{0 < \eta(1 - \beta)\lambda < 2(1 + \beta)}. \quad (4.11)$$

Remark

Compare this to plain gradient descent ($\beta = 0$), where the stability condition is $0 < \eta\lambda < 2$. Momentum *widens* the stable learning-rate range by the factor $(1 + \beta)/(1 - \beta)$. With $\beta = 0.9$, the stable range is $19\times$ wider than without momentum—though the effective step also scales, so the benefit is in convergence speed, not just stability.

Three convergence regimes. The discriminant $a_\lambda^2 - 4\beta$ determines the character of the roots:

- **Overdamped** ($a_\lambda^2 > 4\beta$, both roots real and positive): smooth monotonic decay, but the slower root limits the rate.
- **Critically damped** ($a_\lambda^2 = 4\beta$, repeated root $q = a_\lambda/2$): classically the fastest nonoscillatory settling for this scalar mode.
- **Underdamped** ($a_\lambda^2 < 4\beta$, complex-conjugate pair): oscillatory decay with envelope $(\sqrt{\beta})^t$. (The product of the roots is β , so each has modulus $\sqrt{\beta}$.)

The repeated-root boundaries occur at

$$\eta(1 - \beta)\lambda = (1 - \sqrt{\beta})^2 \quad \text{and} \quad \eta(1 - \beta)\lambda = (1 + \sqrt{\beta})^2. \quad (4.12)$$

At $\beta = 0$, the roots reduce to $\{a_\lambda, 0\}$ and the two boundaries coalesce; the overdamped/underdamped distinction disappears.

Design challenge A single learning rate η must handle every eigenvalue λ in the spectrum simultaneously. A small eigenvalue wants a large η to escape the overdamped regime; a large eigenvalue may already be underdamped or near the stability boundary. Tuning η and β is therefore a minimax problem: make the worst-case convergence rate $\max_i |q_\pm(\lambda_i)|$ as small as possible. The lower panel of Figure 4.2 locates these root regimes for one scalar mode.

Remark

The trajectory comparison in Figure 4.3 illustrates filtering; filtering and acceleration are related but not identical claims. Tuned Nesterov acceleration has a standard $\sqrt{\kappa}$ iteration dependence for smooth strongly convex objectives—strongly convex means curvature bounded below by a positive constant, and smooth means gradient slope bounded above. Tuned Polyak heavy-ball has an analogous rate for quadratics. A generic exponential moving average does not automatically inherit either guarantee.

Nesterov momentum evaluates the gradient where the current momentum predicts the iterate will go. One complete gradient-direction convention is

$$\begin{aligned} v_{t+1} &= \beta v_t + \nabla f(w_t - \alpha \beta v_t), \\ w_{t+1} &= w_t - \alpha v_{t+1}. \end{aligned} \quad (4.13)$$

Other common forms reverse the sign of the velocity or absorb coefficients into the learning rate. Compare the complete recurrence, not one line taken from a different convention.

4.2 Adam

The implicit regularization of gradient descent is both a feature and a bug. It helps with not overfitting to noise by early stopping, but also leads to slow convergence in the directions of smaller

(a) Filter modes and convention matching

input mode g_t	normalized EMA $z_{t+1} = \beta z_t + (1 - \beta)g_t$	unnormalized velocity $v_{t+1} = \beta v_t + g_t$
persistent $g_t = c$	$z_\infty / c = 1$	$v_\infty / c = 1/(1 - \beta)$
alternating $g_t = (-1)^t c$	$ z / c = (1 - \beta)/(1 + \beta)$	$ v / c = 1/(1 + \beta)$

Match effective steps before comparing nominal rates. $\alpha_v = \eta_z(1 - \beta)$.

(b) Scalar-quadratic roots and trajectories

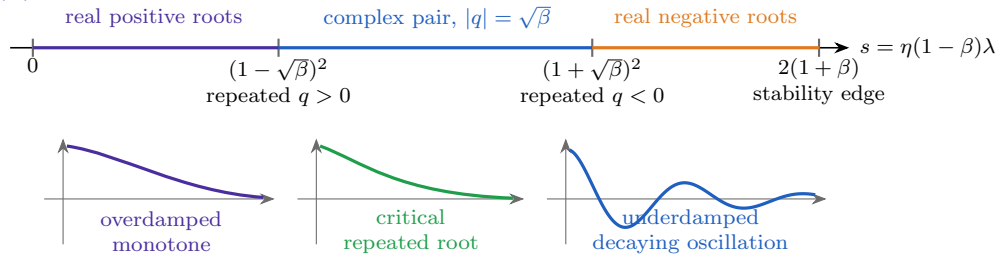


Figure 4.2: The same convention must be used in the filter and optimization views. The upper ledger compares steady modal gains after matching effective parameter steps. The lower panel locates the scalar-quadratic root regimes. Here “critical” marks the repeated-root boundary of this scalar mode.

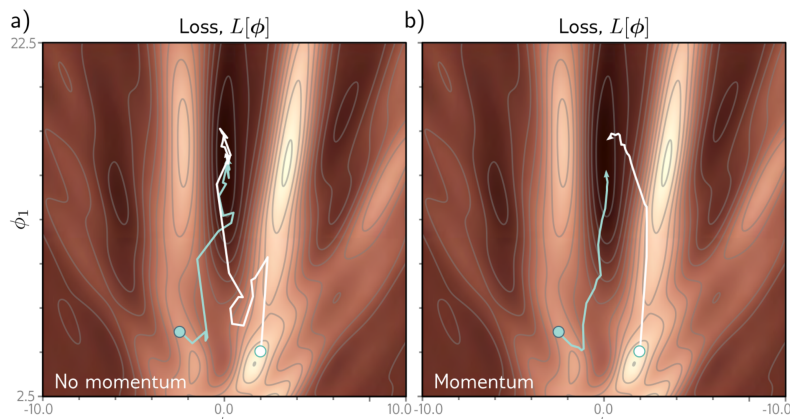


Figure 4.3: A momentum illustration adapted from Prince, Figure 6.7. Exponential averaging suppresses rapid cross-valley oscillation and can yield a smoother trajectory; the acceleration claim still requires the objective and tuning assumptions stated above.

singular values — those that are not so small that you want to ignore them, but still much smaller than the strongest singular values. So there can be a benefit to having steps that are normalized along all the singular directions. However, this would require computing the SVD and then normalizing all directions by the appropriate singular value, a prohibitively expensive operation.

So Adam (and sign SGD) try to do the next best thing. Can we have the steps in each direction in the nominal coordinate system (i.e. not the SVD basis) be roughly the same? Perhaps this can help mitigate the rapid progress in some directions and slow progress in others.

Let $g_t = \nabla \mathcal{L}(w_t)$ or a mini-batch estimate. If one sets

$$m_{t+1} = g_t, \quad v_{t+1} = g_t \odot g_t, \quad (4.14)$$

then, ignoring a stabilizer and restricting to coordinates for which $g_{t,j} \neq 0$,

$$\frac{m_{t+1}}{\sqrt{v_{t+1}}} = \text{sgn}(g_t) \quad (4.15)$$

coordinatewise. This instantaneous coordinate normalization gives us SignSGD (when gradients are non-zero). However, relying entirely on the instantaneous gradient direction can cause the optimizer to overshoot and bounce back and forth across a narrow valley once it is close to the minimum.

To smooth out this trajectory, Adam introduces exponential moving averages with decay parameters $0 \leq \beta_1, \beta_2 < 1$ for both the numerator and denominator:

$$m_{t+1} = \beta_1 m_t + (1 - \beta_1) g_t, \quad (4.16)$$

$$v_{t+1} = \beta_2 v_t + (1 - \beta_2)(g_t \odot g_t), \quad m_0 = v_0 = 0. \quad (4.17)$$

Because they are initialized to zero, these moment estimates are biased toward zero early in training. For example, with $\beta_1 = 0.9$, the first momentum update is

$$m_1 = 0.9(0) + 0.1g_0. \quad (4.18)$$

This estimate is too small (it is $0.1g_0$ rather than g_0) and can lead to a slow start. To correct this bias, Adam scales up the early moments. With the $t+1$ indexing above, the corrected estimates are

$$\hat{m}_{t+1} = \frac{m_{t+1}}{1 - \beta_1^{t+1}}, \quad \hat{v}_{t+1} = \frac{v_{t+1}}{1 - \beta_2^{t+1}}. \quad (4.19)$$

With a stabilizer $\varepsilon > 0$, Adam then updates

$$w_{t+1} = w_t - \eta \frac{\hat{m}_{t+1}}{\sqrt{\hat{v}_{t+1} + \varepsilon}}. \quad (4.20)$$

All products, roots, and divisions in Equations (4.17)–(4.20) are coordinatewise.

Adam combines a momentum-like first moment with a coordinatewise second-moment scale. It can make the effective step less sensitive to raw gradient magnitude and can remain useful when different coordinates have very different scales. Its optimizer state is correspondingly larger: in addition to parameters, it stores both m_t and v_t .

The contrast is visible in Figure 4.4. In this narrow valley, fixed coordinate-normalized steps repeatedly cross the valley near the minimizer, while Adam's accumulated moments bend and smooth the trajectory.

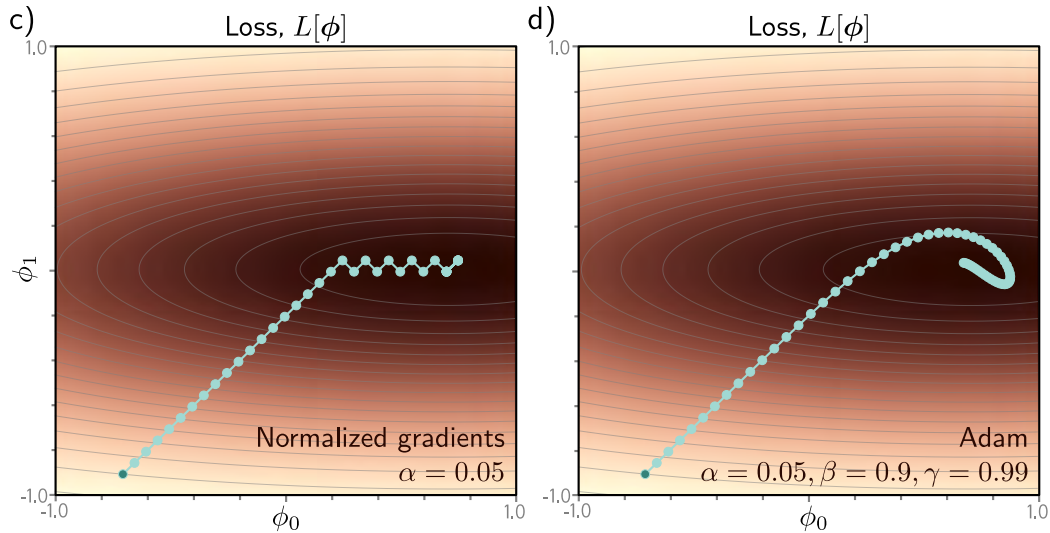


Figure 4.4: The coordinate-normalized update, which reduces to SignSGD away from zero-gradient coordinates, enters a two-point oscillation near the minimum; Adam follows a smoother trajectory for the displayed hyperparameters. From Simon J. D. Prince, *Understanding Deep Learning*, Figure 6.9(c)–(d).

Remark

Quantizing optimizer or gradient state can trade numerical fidelity for lower storage or communication. Optimizer state is still only one part of the full memory and communication ledger. Parameters, master copies, gradients, activations, workspace, and communication buffers can also matter.

Remark

Why should losing raw magnitude information help? We do not know one universal explanation. Turns out there were small issues in the original Adam paper. Subsequent analyses and variants established convergence guarantees under additional conditions. Meanwhile the practical method remained useful and its defaults often worked. The evidence supports Adam as a robust engineering tool, while problem families can favor Adam or momentum SGD. The update can keep “steering towards the right thing” because a later gradient measures the state that actually resulted and can correct the next move. This is feedback intuition, not a convergence proof; the lower panel of Figure 4.5 depicts that distinction.

4.3 AdamW and decoupled weight decay

For ordinary gradient descent, adding ridge regularization

$$\mathcal{L}_\lambda(w) = \mathcal{L}(w) + \lambda \|w\|_2^2 \quad (4.21)$$

gives

$$w_{t+1} = (1 - 2\eta\lambda)w_t - \eta\nabla\mathcal{L}(w_t). \quad (4.22)$$

The ridge gradient and multiplicative shrinkage are equivalent in this plain-GD setting.

If the ridge gradient is inserted into Adam, however, it enters both moment accumulators and is rescaled coordinatewise along with the data-loss gradient. AdamW instead applies shrinkage as a separate operation:

$$w_{t+1} = (1 - \eta\lambda_{\text{wd}})w_t - \eta \frac{\hat{m}_{t+1}}{\sqrt{\hat{v}_{t+1} + \varepsilon}}, \quad (4.23)$$

where the moments are formed from the unregularized data-loss gradient. The coefficient convention matters: λ_{wd} in Equation (4.23) is the rate inside the factor $(1 - \eta\lambda_{\text{wd}})$.

These methods sit on a historical arc from classical gradient descent and mid-century stochastic approximation to modern adaptive methods. The engineering lesson is more important than the chronology: an optimizer seeks a low objective value, while deep learning ultimately needs a solution that generalizes. Neither Adam nor momentum SGD is uniformly best across problems.

For a quadratic, separate learning rates in the Hessian eigendirections would directly address the curvature. Those directions are generally unavailable in a large nonlinear model. Parameter and layer coordinates are cheap and visible, which is why practical methods organize their state there even when those coordinates do not diagonalize the local geometry.

Optimizer choice therefore has at least three axes. Ask how quickly and reliably the training objective falls, which solutions and generalization behavior the method favors, and what memory, communication, and throughput the implementation costs. None of these axes supplies a universal ordering. The upper ledger in Figure 4.5 keeps the three comparisons separate.

(a) Three ledgers for an optimizer comparison

method	training convergence	generalization / implicit bias	memory / throughput
SGD	raw g_t ; rate depends on geometry and tuning	update span and stochastic path; measure on target task	no extra optimizer vector
momentum	filtered first moment; pole placement matters	different trajectory and stopping behavior; measure	one parameter-sized state vector
Adam	first and second moments; coordinate scaling	preconditioned path differs from SGD; measure	two parameter-sized state vectors

These are local optimizer-state counts, not a complete training-memory ledger. No row is a universal winner.

(b) Feedback makes later correction possible

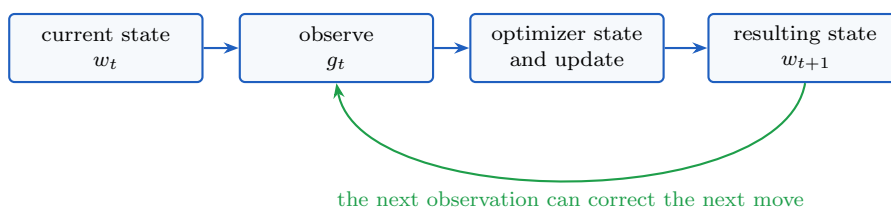


Figure 4.5: An optimizer comparison is a three-axis ledger, not a single rank. SGD, momentum, and Adam expose different update state and geometry. Their convergence and generalization behavior remain problem-dependent, while the systems column records symbolic state costs rather than a complete memory budget. The inset shows why a useful update may be corrected by later feedback instead of aligning perfectly with one instantaneous gradient.

Recap

- Momentum is a low-pass filter on stochastic gradients; its stability on each eigenmode is governed by a second-order recurrence.
- The Schur criterion gives the exact stability region; tuning across the full Hessian spectrum is a minimax problem.
- Adam smooths first and second coordinatewise moments and corrects their initialization bias; it is only SignSGD-like.
- AdamW separates multiplicative weight decay from Adam’s adaptive moment machinery.

Further reading

- Qian (1999). Analyzes the momentum recursion as a coupled dynamical system, the basis for the eigenmode analysis in this note.
- Kingma and Ba (2015). The original Adam algorithm, including moment estimates and bias correction.
- Loshchilov and Hutter (2019). Explains why ℓ_2 regularization and weight decay cease to be equivalent under adaptive preconditioning and proposes decoupling.

References and Further Reading

- Kingma, D. P. and J. Ba (2015). “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations*. arXiv: 1412.6980. URL: <https://arxiv.org/abs/1412.6980>.
- Loshchilov, I. and F. Hutter (2019). “Decoupled Weight Decay Regularization”. In: *7th International Conference on Learning Representations*. arXiv: 1711.05101. URL: <https://arxiv.org/abs/1711.05101>.
- Qian, N. (1999). “On the Momentum Term in Gradient Descent Learning Algorithms”. In: *Neural Networks* 12.1, pp. 145–151. DOI: 10.1016/S0893-6080(98)00116-6. URL: <https://www.columbia.edu/~nq6/publications/momentum.pdf>.