In the Adam and AdamW optimizers, the bias corrections for the first moment estimate $ m_t $ (mean of gradients) and the second raw moment estimate $ v_t $ (uncentered variance of gradients) are necessary to address the initialization bias of the exponential moving averages. These corrections ensure that the estimates are unbiased, especially in the early stages of training. Let's break down why $ m_t $ is divided by $ (1 - \beta_1^t) $ and $ v_t $ by $ (1 - \beta_2^t) $.

Background

Adam (and its variant AdamW) maintains two moving averages:

First moment ($ m_t $): Tracks the exponentially decaying average of past gradients, controlled by the hyperparameter $ \beta_1 $.

Second raw moment ($ v_t $): Tracks the exponentially decaying average of the squared gradients, controlled by $ \beta_2 $.

These are updated as follows:

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

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

where $ g_t $ is the gradient at time step $ t $.

At initialization ($ t = 0 $), $ m_0 = 0 $ and $ v_0 = 0 $. This zero initialization introduces a bias, causing $ m_t $ and $ v_t $ to underestimate the true mean and variance of the gradients, especially in early iterations when $ t $ is small and the moving averages have not yet accumulated enough information.

Why Bias Correction?

The bias correction terms $ \frac{m_t}{1 - \beta_1^t} $ and $ \frac{v_t}{1 - \beta_2^t} $ adjust for this initialization bias, ensuring that $ m_t $ and $ v_t $ approximate the expected values of the gradient mean and squared gradient variance, respectively, as if the moving averages had been running for an infinite number of steps.

1. Bias Correction for $ m_t $: Dividing by $ (1 - \beta_1^t) $

Reason: The first moment $ m_t $ is an exponentially weighted moving average of past gradients. Because $ m_0 = 0 $, early values of $ m_t $ are biased toward zero, especially when $ \beta_1 $ (e.g., 0.9) is large, as the

contribution of past gradients decays slowly. Dividing by $ (1 - \beta_1^t) $ corrects this bias by scaling $ m_t $ to estimate the true expected gradient.
Mathematical Intuition:

The update rule for $ m_t $ is:

$$m_t = (1 - \beta_1) \sum_{i=1}^{t} \beta_1^{t-i} g_i$$

The weights $ \beta_1^{t-i} $ sum to:

$$\sum_{i=1}^{t} (1 - \beta_1) \beta_1^{t-i} = (1 - \beta_1) \cdot \frac{1-\beta_1^t}{1-\beta_1} = 1 - \beta_1^t$$

Since the weights do not sum to 1 (they sum to $ 1 - \beta_1^t $), $ m_t $ is a biased estimate of the gradient mean. Dividing by $ 1 - \beta_1^t $ normalizes the weights to sum to 1:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

This gives an unbiased estimate of the expected gradient, assuming gradients are stationary.

Example: Suppose $ \beta_1 = 0.9 $, and at $ t = 1 $, $ g_1 = 1 $. Then:

$$m_1 = \beta_1 \cdot 0 + (1 - \beta_1) \cdot 1 = 0.1$$

Without correction, $ m_1 = 0.1 $, which underestimates the true gradient (1).
The bias correction is:

$$\hat{m}_1 = \frac{0.1}{1-0.9^1} = \frac{0.1}{0.1} = 1$$

This matches the true gradient, correcting the bias introduced by the zero initialization.

1. Bias Correction for $ v_t $: Dividing by $ (1 - \beta_2^t) $

Reason: Similarly, the second moment $ v_t $ is an exponentially weighted moving average of squared gradients, initialized at $ v_0 = 0 $. This causes $ v_t $ to underestimate the true variance of the gradients in early iterations. Dividing by $ (1 - \beta_2^t) $ corrects this bias to approximate the expected squared gradient.
Mathematical Intuition:

The update rule for $ v_t $ is:

$$v_t = (1 - \beta_2) \sum_{i=1}^{t} \beta_2^{t-i} g_i^2$$

The weights sum to $ 1 - \beta_2^t $, just like in the first moment case.

Dividing by $ 1 - \beta_2^t $ normalizes the weights:
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$
This provides an unbiased estimate of the expected squared gradient.

Example: Suppose $ \beta_2 = 0.999 $, and at $ t = 1 $, $ g_1^2 = 1 $. Then:
$$v_1 = \beta_2 \cdot 0 + (1 - \beta_2) \cdot 1 = 0.001$$
Without correction, $ v_1 = 0.001 $, which underestimates the true squared gradient (1). The bias correction is:
$$\hat{v}_1 = \frac{0.001}{1-0.999^1} = \frac{0.001}{0.001} = 1$$
This corrects the estimate to match the true squared gradient.

Why Is This Important?

Early Training Stability: In the early steps ($ t $ small), $ \beta_1^t $ and $ \beta_2^t $ are significant (e.g., for $ \beta_1 = 0.9 $, $ \beta_1^1 = 0.9 $, $ \beta_1^2 = 0.81 $), making the bias substantial. Without correction, the optimizer would take smaller or incorrect steps, slowing convergence or causing instability (e.g., loss spikes in LLM training).
AdamW Specificity: AdamW modifies Adam by decoupling weight decay from the optimization step, but the bias correction for $ m_t $ and $ v_t $ remains identical, as it is a core part of the Adam algorithm.
Practical Impact: In LLM training, uncorrected estimates could lead to overly small updates (due to underestimating $ m_t $ or $ v_t $), delaying convergence, or overly large updates (if $ v_t $ is too small), causing loss spikes.

Summary

Dividing $ m_t $ by $ 1 - \beta_1^t $ corrects the bias in the first moment estimate caused by zero initialization, ensuring $ \hat{m}*t $ approximates the true gradient mean.*

*Dividing $ v_t $ by $ 1 - |beta_2^t $ corrects the bias in the second moment estimate, ensuring $ |hat{v}*t $ approximates the true squared gradient variance.

These corrections are critical in early training to ensure stable and effective parameter updates, especially for large-scale models like LLMs where training stability is crucial.