CPEN 455: Deep Learning
Backpropagation and Initialization

Companion notes to Lecture 4.1

Renjie Liao
With assistance from ChatGPT-6 Astra Ultra.
University of British Columbia Winter Term 1, 2026

1 Gradient-based Learning and the Forward Pass

Gradient-based learning fits parameters by reducing a training objective. Stochastic gradient methods use sampled data to estimate its gradient [1]. *Backpropagation* computes the derivatives needed for an update by applying the chain rule through the network [2]. The optimizer then uses those derivatives to change the parameters. Computing a gradient and taking a learning step are separate operations.

Network and notation

Consider an MLP with M hidden layers. Layer i has width N_i, the input has dimension N_0, and the output has dimension N_{M+1}. Vectors are columns; bold symbols denote vectors and W_i denotes a weight matrix. Brackets select coordinates or entries, e.g. $\mathbf{h}_i[j]$ or $W_i[j,k]$. To match the slides' first derivation, omit biases for now:

$$\mathbf{h}_0 = \mathbf{x}, \qquad \mathbf{z}_i = W_i \mathbf{h}_{i-1}, \qquad \mathbf{h}_i = \sigma(\mathbf{z}_i), \quad i = 1, \dots, M, \tag{1}$$

$$\mathbf{y} = W_{M+1} \mathbf{h}_M, \qquad L = \ell(\mathbf{y}, \bar{\mathbf{y}}). \tag{2}$$

The activation σ acts on each coordinate. $\mathbf{z}_i$ is the *preactivation*; $\mathbf{h}_i$ is the activation. The output layer is linear. Here $\mathbf{y}$ is the prediction and $\bar{\mathbf{y}}$ is the fixed target, as in the original BP slides. The loss L is scalar.

$$W_i \in \mathbb{R}^{N_i \times N_{i-1}}, \qquad \mathbf{z}_i, \mathbf{h}_i \in \mathbb{R}^{N_i \times 1}, \qquad \mathbf{y}, \bar{\mathbf{y}} \in \mathbb{R}^{N_{M+1} \times 1}.$$

The slides also write the output width as $N_y = N_{M+1}$. The weight-matrix shape applies at $i = M+1$. The layer index is i in this derivation, m in the bias and mini-batch slides, and k in initialization; these indices play the same role. Biases are added in Section 3.

Loss and learning

For one example, a possible regression loss and its output gradient are

$$L = \frac{1}{2} \|\mathbf{y} - \bar{\mathbf{y}}\|_2^2, \qquad \frac{\partial L}{\partial \mathbf{y}} = \mathbf{y} - \bar{\mathbf{y}}.$$

Backpropagation also works with other differentiable losses: only the starting output gradient changes. A classification loss can accept $\mathbf{y}$ as logits and include the output transformation internally, as in the Lecture 2 notes.

For a mini-batch of B examples, use

$$L_{\mathcal{B}} = \frac{1}{B} \sum_{b=1}^{B} \ell(\mathbf{y}^{(b)}, \bar{\mathbf{y}}^{(b)}).$$

First derive the gradients for one example, then average them. During a backward pass, all parameters are held at the values used in the forward pass.

2 Gradients, Jacobians, and the Chain Rule

For a scalar loss and $\mathbf{x} \in \mathbb{R}^{n\times 1}$, derivatives with respect to vectors are *column gradients*:

$$\frac{\partial L}{\partial \mathbf{x}} = \nabla_{\mathbf{x}} L = \left[\frac{\partial L}{\partial \mathbf{x}[1]}, \ldots, \frac{\partial L}{\partial \mathbf{x}[n]}\right]^{\top}.$$

For $\mathbf{y} = g(\mathbf{x}) \in \mathbb{R}^{m\times 1}$, the Jacobian has output coordinates in its rows and input coordinates in its columns:

$$\mathbf{J}_g[i,j] = \frac{\partial \mathbf{y}[i]}{\partial \mathbf{x}[j]}, \qquad \mathbf{J}_g \in \mathbb{R}^{m\times n}.$$

Changing $\mathbf{x}[j]$ affects the loss through each output coordinate. Summing these contributions gives the chain rule:

$$\frac{\partial L}{\partial \mathbf{x}[j]} = \sum_{i=1}^{m} \frac{\partial L}{\partial \mathbf{y}[i]} \frac{\partial \mathbf{y}[i]}{\partial \mathbf{x}[j]} \quad \Longrightarrow \quad \boxed{\frac{\partial L}{\partial \mathbf{x}} = \mathbf{J}_g^{\top} \frac{\partial L}{\partial \mathbf{y}}.} \tag{3}$$

The transpose gives the required shape: $(n \times m)(m \times 1) = n \times 1$.

Local Jacobians and the final output

For a hidden layer, the slides define

$$\mathbf{J}_i := \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} = \operatorname{diag}\big(\sigma'(\mathbf{z}_i)\big) W_i \in \mathbb{R}^{N_i \times N_{i-1}}, \qquad i = 1, \ldots, M.$$

For the linear output $\mathbf{y} = W_{M+1}\mathbf{h}_M$, the gradient at the final hidden activation is

$$\frac{\partial L}{\partial \mathbf{h}_M} = W_{M+1}^{\top} \frac{\partial L}{\partial \mathbf{y}}.$$

The last Jacobian in the compact slide formula

In the compact chain, the terminal $\mathbf{J}_M$ includes the output map. To distinguish this combined factor from the hidden-layer-only $\mathbf{J}_M$ defined above, write

$$\begin{aligned} \mathbf{J}_M^{\text{last}} &:= \frac{\partial \mathbf{y}}{\partial \mathbf{h}_{M-1}} = W_{M+1} \operatorname{diag}\big(\sigma'(\mathbf{z}_M)\big) W_M \\ &= W_{M+1}\mathbf{J}_M \in \mathbb{R}^{N_{M+1}\times N_{M-1}}. \end{aligned}$$

It differentiates the whole final block $\mathbf{h}_{M-1} \mapsto \mathbf{h}_M \mapsto \mathbf{y}$. Thus, for $0 \le i < M$,

$$\frac{\partial L}{\partial \mathbf{h}_i} = \mathbf{J}_{i+1}^{\top} \cdots \mathbf{J}_{M-1}^{\top} (\mathbf{J}_M^{\text{last}})^{\top} \frac{\partial L}{\partial \mathbf{y}}. \tag{4}$$

Since $(\mathbf{J}_M^{\text{last}})^{\top} = \mathbf{J}_M^{\top} W_{M+1}^{\top}$, the output factor is absorbed into the terminal Jacobian. At $i = M-1$, the preceding product is empty. The gradient at $\mathbf{h}_M$ is given separately above; at $i = 0$, use $\mathbf{h}_0 = \mathbf{x}$. Backpropagation evaluates the chain from right to left, reusing the gradient from the next block.

3 Weight and Bias Gradients

Why a weight gradient is an outer product

In the output-weight derivation, the slides abbreviate $\mathbf{h}_M$ as $\mathbf{h}$. For the linear output, $\mathbf{y}[i] = \sum_j W_{M+1}[i,j]\mathbf{h}_M[j]$. Changing $W_{M+1}[i,j]$ directly changes only coordinate $\mathbf{y}[i]$, so

$$\frac{\partial L}{\partial W_{M+1}[i,j]} = \frac{\partial L}{\partial \mathbf{y}[i]}\mathbf{h}_M[j], \qquad \frac{\partial L}{\partial W_{M+1}} = \frac{\partial L}{\partial \mathbf{y}}\mathbf{h}_M^\top.$$

The matrix gradient is defined entry by entry and has the same shape as its weight matrix. This component derivation avoids treating a derivative with respect to a matrix as an ordinary two-dimensional Jacobian.

Adding biases and defining the local gradient

As in the bias slides, use $\mathbf{z}_m = W_m\mathbf{h}_{m-1} + \mathbf{b}_m$, $\mathbf{h}_m = \sigma_m(\mathbf{z}_m)$, and $\mathbf{y} = W_{M+1}\mathbf{h}_M + \mathbf{b}_{M+1}$. Here $\mathbf{b}_m \in \mathbb{R}^{N_m \times 1}$; σ_m allows a different activation in each layer. The target written $\mathbf{g}$ there is the same $\bar{\mathbf{y}}$ used above: $L = \ell(\mathbf{y}, \mathbf{g}) = \ell(\mathbf{y}, \bar{\mathbf{y}})$.

Set $\mathbf{z}_{M+1} := \mathbf{y}$ for the linear output. With the slides' gradient notation, $\boldsymbol{\delta}_m := \nabla_{\mathbf{z}_m} L = \partial L/\partial \mathbf{z}_m$ for hidden layers and $\boldsymbol{\delta}_{M+1} := \nabla_{\mathbf{y}} L$. The backward recursion is

$$\boxed{\boldsymbol{\delta}_m = (W_{m+1}^\top \boldsymbol{\delta}_{m+1}) \odot \sigma_m'(\mathbf{z}_m), \qquad m = M, \ldots, 1.} \tag{5}$$

Here $\odot$ is elementwise multiplication. The transposed weight matrix propagates sensitivity from the next layer; $\sigma_m'(\mathbf{z}_m)$ accounts for the local nonlinearity. Biases affect the saved preactivations, but the same backward recursion still applies.

For a matrix, $\nabla_{W_m} L$ denotes the entrywise gradient $\partial L/\partial W_m$. The component equation

$$\mathbf{z}_m[i] = \sum_j W_m[i,j]\mathbf{h}_{m-1}[j] + \mathbf{b}_m[i]$$

gives $\partial \mathbf{z}_m[i]/\partial W_m[i,j] = \mathbf{h}_{m-1}[j]$ and $\partial \mathbf{z}_m[i]/\partial \mathbf{b}_m[i] = 1$. Including the linear output layer,

$$\boxed{\nabla_{W_m} L = \boldsymbol{\delta}_m \mathbf{h}_{m-1}^\top, \qquad \nabla_{\mathbf{b}_m} L = \boldsymbol{\delta}_m, \qquad m = 1, \ldots, M+1.} \tag{6}$$

An $N_m \times 1$ column times a $1 \times N_{m-1}$ row produces exactly the $N_m \times N_{m-1}$ entries needed for W_m. The bias gradient has N_m entries.

One training step

1. Run the forward pass and save the activations and preactivations.
2. Evaluate the loss and its output gradient $\boldsymbol{\delta}_{M+1}$.
3. Work backward using (5) and (6).
4. After all gradients are computed, apply the optimizer update.

For gradient descent, $W_m \leftarrow W_m - \eta \nabla_{W_m} L$ and $\mathbf{b}_m \leftarrow \mathbf{b}_m - \eta \nabla_{\mathbf{b}_m} L$. For an activation that is not differentiable at a point, an implementation must specify a derivative convention there, such as a subgradient at a ReLU kink.

4 Mini-batches and Efficient Computation

Average the parameter gradients once

Let $\ell^{(b)} = \ell(\mathbf{y}^{(b)}, \bar{\mathbf{y}}^{(b)})$. Define each $\boldsymbol{\delta}_m^{(b)}$ using the *per-example loss* $\ell^{(b)}$, not the averaged batch loss. Differentiation is linear, so

$$\nabla_{W_m} L_{\mathcal{B}} = \frac{1}{B} \sum_{b=1}^{B} \boldsymbol{\delta}_m^{(b)} (\mathbf{h}_{m-1}^{(b)})^\top, \qquad \nabla_{\mathbf{b}_m} L_{\mathcal{B}} = \frac{1}{B} \sum_{b=1}^{B} \boldsymbol{\delta}_m^{(b)}. \tag{7}$$

Stack examples as *columns*, as in the BP slides:

$$H_{m-1} = [\mathbf{h}_{m-1}^{(1)}, \ldots, \mathbf{h}_{m-1}^{(B)}] \in \mathbb{R}^{N_{m-1} \times B}, \qquad \Delta_m = [\boldsymbol{\delta}_m^{(1)}, \ldots, \boldsymbol{\delta}_m^{(B)}] \in \mathbb{R}^{N_m \times B}.$$

With $\mathbf{1} \in \mathbb{R}^{B \times 1}$ containing ones,

$$\boxed{\nabla_{W_m} L_{\mathcal{B}} = \frac{1}{B} \Delta_m H_{m-1}^\top, \qquad \nabla_{\mathbf{b}_m} L_{\mathcal{B}} = \frac{1}{B} \Delta_m \mathbf{1}.} \tag{8}$$

Lecture 3's BatchNorm discussion stores examples as rows; that convention is equivalent after transposing the batch representation. If the backward pass starts from $L_{\mathcal{B}}$ itself, its local gradients already contain $1/B$: do not divide by B a second time.

The per-example recursion assumes that examples do not interact in the forward pass. Training-time BatchNorm couples examples through batch statistics, so its backward pass must differentiate those shared statistics. Backpropagation still applies, but treating each example as an independent network is insufficient.

Reuse the forward computation

The backward pass needs the saved $\mathbf{h}_{m-1}$ for weight gradients and the information needed to evaluate $\sigma'_m(\mathbf{z}_m)$. It computes products such as $W_m^\top \boldsymbol{\delta}_m$ and $\boldsymbol{\delta}_m \mathbf{h}_{m-1}^\top$ directly; there is no need to build a full diagonal activation Jacobian or the long product in (4). For a dense layer, these two main backward products each have a cost comparable to the forward multiplication. This explains the rough "twice the forward work" estimate; actual runtime depends on the operations and hardware. Saving intermediate values trades memory for less recomputation.

Automatic differentiation and branching graphs

Backpropagation is reverse-mode automatic differentiation [4]. Traverse operations in reverse order, multiply by local derivatives, and *add* contributions whenever a value is used by several later operations. The AD slide writes a local gradient as an *adjoint*:

$$\bar{\mathbf{v}} := \nabla_{\mathbf{v}} L = \frac{\partial L}{\partial \mathbf{v}}, \qquad \bar{\mathbf{v}} = \sum_{\mathbf{u}:\, \mathbf{v} \to \mathbf{u}} \mathbf{J}_{\mathbf{u},\mathbf{v}}^\top \bar{\mathbf{u}}, \quad \mathbf{J}_{\mathbf{u},\mathbf{v}} := \frac{\partial \mathbf{u}}{\partial \mathbf{v}}.$$

Here the overbar means a backward gradient. Elsewhere, $\bar{\mathbf{y}}$ denotes the fixed target; these are separate uses of the overbar. Shared parameters likewise receive the sum of contributions from all their uses. Automatic differentiation applies analytic local rules to the executed computation; it does not estimate derivatives by perturbing every parameter.

5 Initialization and Forward Variance

Symmetry and scale

Hidden units with identical incoming parameters and identical outgoing connections compute the same features and receive identical updates under deterministic gradient descent. Independent random weights break this symmetry. Biases can still start at zero when the weights differ.

Randomness alone does not choose a useful scale. Very small signals can shrink through depth; very large preactivations can saturate tanh and make its derivative small. Initialization aims to keep both forward signals and backward gradients at useful scales, before training changes the weights.

The variance identities used in the slides

The notation $\mathbb{V}[\cdot]$ is the same as $\mathrm{Var}(\cdot)$ in the assumptions slide. For a scalar random variable u with finite second moment,

$$\mathbb{V}[u] = \mathbb{E}[(u - \mathbb{E}[u])^2] = \mathbb{E}[u^2] - \mathbb{E}[u]^2.$$

Expectations add even without independence. For a sum,

$$\mathbb{V}\left[\sum_j u_j\right] = \sum_j \mathbb{V}[u_j] + 2\sum_{i<j} \mathrm{Cov}(u_i, u_j).$$

The covariance terms vanish for uncorrelated summands, in particular independent ones. If w and h are independent and $\mathbb{E}[w] = 0$, then

$$\mathbb{E}[wh] = 0, \qquad \mathbb{V}[wh] = \mathbb{E}[w^2]\mathbb{E}[h^2] = \mathbb{V}[w]\mathbb{E}[h^2]. \tag{9}$$

Only when h is also centered can $\mathbb{E}[h^2]$ be replaced by $\mathbb{V}[h]$. Appendix A records the general product-variance formula.

A simplified initialization model

Use zero biases and independent, zero-mean weights. Assume centered input coordinates and tanh preactivations near zero, so $\tanh(z) \approx z$ and $\tanh'(z) \approx 1$. Weight and activation variances are equal across coordinates *within each layer*, but may differ between layers. For backward analysis, also approximate gradients as centered and neglect their dependence on weights. These are initialization approximations, not exact properties during training.

The linearized forward pass and (9) give

$$\begin{aligned}
\mathbf{h}_k[i] &\approx \sum_{j=1}^{N_{k-1}} W_k[i,j]\mathbf{h}_{k-1}[j], \\
\mathbb{V}[\mathbf{h}_k[i]] &\approx \sum_{j=1}^{N_{k-1}} \mathbb{V}[W_k[i,j]]\mathbb{V}[\mathbf{h}_{k-1}[j]] \\
&= N_{k-1}\mathbb{V}[W_k[i,j]]\mathbb{V}[\mathbf{h}_{k-1}[j]].
\end{aligned} \tag{10}$$

Independent centered weights make distinct summands uncorrelated under this model. Thus the *fan-in* choice $\mathbb{V}[W_k[i,j]] = 1/N_{k-1}$ keeps forward variance approximately unchanged. As in the slides, coordinate indices in these variance formulas represent any valid entries of their respective layers.

6 Backward Variance and Xavier Initialization

How scale compounds through depth

Unrolling (10) through hidden layer l gives

$$\mathbb{V}[\mathbf{h}_l[i]] \approx \mathbb{V}[\mathbf{x}[j]] \prod_{k=1}^{l} N_{k-1}\mathbb{V}[W_k[i,j]].$$

Products of factors below or above one can shrink or enlarge the signal rapidly. For backward gradients, the linearized chain rule is

$$\frac{\partial L}{\partial \mathbf{h}_{k-1}[j]} \approx \sum_{i=1}^{N_k} W_k[i,j]\frac{\partial L}{\partial \mathbf{h}_k[i]}.$$

At the output, interpret $\mathbf{h}_{M+1}$ as $\mathbf{y}$. With the centering and independence approximations from Section 5,

$$\begin{aligned}\mathbb{V}\left[\frac{\partial L}{\partial \mathbf{h}_{k-1}[j]}\right] &\approx N_k\mathbb{V}[W_k[i,j]]\mathbb{V}\left[\frac{\partial L}{\partial \mathbf{h}_k[i]}\right], \\ \mathbb{V}\left[\frac{\partial L}{\partial \mathbf{h}_l[i]}\right] &\approx \mathbb{V}\left[\frac{\partial L}{\partial \mathbf{y}[i]}\right] \prod_{k=l+1}^{M+1} N_k\mathbb{V}[W_k[i,j]].\end{aligned} \tag{11}$$

The *fan-out* choice $\mathbb{V}[W_k[i,j]] = 1/N_k$ keeps backward variance approximately unchanged. Actual gradients depend on the forward weights, so this factorization is an approximation.

Balancing forward and backward propagation

The forward and backward choices are, respectively,

$$\mathbb{V}[W_k[i,j]] = \frac{1}{N_{k-1}}, \qquad \mathbb{V}[W_k[i,j]] = \frac{1}{N_k}.$$

They agree when the widths match. Otherwise, a single variance cannot make both propagation multipliers equal one. Xavier/Glorot initialization uses the average of the denominators [3]:

$$\boxed{\mathbb{V}[W_k[i,j]] = \frac{1}{(N_{k-1}+N_k)/2} = \frac{2}{N_{k-1}+N_k}.} \tag{12}$$

This takes the reciprocal of the average fan count, rather than averaging the two proposed variances. The resulting forward and backward multipliers are

$$\frac{2N_{k-1}}{N_{k-1}+N_k}, \qquad \frac{2N_k}{N_{k-1}+N_k}.$$

Their average is one; each need not equal one. The weight-gradient variance analysis in the slides uses the same assumptions; its derivation is in Appendix A. These initial scale arguments do not guarantee successful optimization for an arbitrary network.

7 Sampling Xavier Weights

The Xavier rule (12) specifies a target variance for zero-mean weights, not a unique distribution. Two common choices draw entries independently as follows. For layer k, the practical slide's notation is $n_{\text{in}} = N_{k-1}$, $n_{\text{out}} = N_k$, and $W_{ij} = W_k[i, j]$.

Gaussian initialization

$$W_k[i, j] \sim \mathcal{N}\left(0, \frac{2}{N_{k-1} + N_k}\right).$$

The second parameter is the *variance*. If a sampler takes a standard deviation instead, use $\sqrt{2/(N_{k-1} + N_k)}$.

Uniform initialization: where the bounds come from

Let $w := W_k[i, j] \sim \mathcal{U}(-a, a)$, where $a > 0$. Its density is $1/(2a)$ on $[-a, a]$ and zero outside. Direct integration gives

$$\mathbb{E}[w] = \frac{1}{2a}\int_{-a}^{a} w\, dw = \frac{1}{2a}\left[\frac{w^2}{2}\right]_{-a}^{a} = 0,$$
$$\mathbb{E}[w^2] = \frac{1}{2a}\int_{-a}^{a} w^2\, dw = \frac{1}{2a}\left[\frac{w^3}{3}\right]_{-a}^{a} = \frac{a^2}{3}.$$

Hence $\mathbb{V}[w] = \mathbb{E}[w^2] - \mathbb{E}[w]^2 = a^2/3$. Matching the Xavier variance yields

$$\frac{a^2}{3} = \frac{2}{N_{k-1} + N_k} \quad \Longrightarrow \quad a = \sqrt{\frac{6}{N_{k-1} + N_k}},$$

so

$$\boxed{W_k[i, j] \sim \mathcal{U}\left(-\sqrt{\frac{6}{N_{k-1} + N_k}}, \sqrt{\frac{6}{N_{k-1} + N_k}}\right).} \tag{13}$$

The Gaussian and uniform choices have the same mean and variance, but they are different distributions. These formulas use gain one and the preceding linearized analysis. For other nonlinearities and normalized or residual architectures, follow the architecture's initialization specification.

Self-check questions

Try these before reading the answers in Appendix B.

1. If $\mathbf{y} = g(\mathbf{x}) \in \mathbb{R}^{3\times 1}$ and $\mathbf{x} \in \mathbb{R}^{5\times 1}$, what are the shapes of $\mathbf{J}_g$ and $\partial L/\partial \mathbf{x}$? Why is a transpose needed?
2. For $W_m \in \mathbb{R}^{4\times 6}$, what are the shapes of $\boldsymbol{\delta}_m$, $\nabla_{W_m} L$, and $\nabla_{\mathbf{b}_m} L$?
3. An automatic backward pass starts from a mean batch loss. Should its parameter gradients be divided by B again? What if a value feeds two branches?
4. Why can random weights break hidden-unit symmetry even when all biases start at zero?
5. Can one variance exactly preserve both forward and backward variances when $N_{k-1} \neq N_k$, under the simplified model?
6. Why is the uniform Xavier bound proportional to $\sqrt{6}$, while the Gaussian standard deviation is proportional to $\sqrt{2}$?

A Further Variance Details

The variance of a product

For independent scalar random variables u, v with finite second moments,

$$\begin{aligned}\mathbb{V}[uv] &= \mathbb{E}[u^2]\mathbb{E}[v^2] - \mathbb{E}[u]^2\mathbb{E}[v]^2 \\ &= (\mathbb{V}[u] + \mathbb{E}[u]^2)(\mathbb{V}[v] + \mathbb{E}[v]^2) - \mathbb{E}[u]^2\mathbb{E}[v]^2 \\ &= \mathbb{V}[u]\mathbb{V}[v] + \mathbb{E}[u]^2\mathbb{V}[v] + \mathbb{E}[v]^2\mathbb{V}[u].\end{aligned}$$

If both means are zero, this reduces to $\mathbb{V}[uv] = \mathbb{V}[u]\mathbb{V}[v]$. If only u is centered, it becomes $\mathbb{V}[uv] = \mathbb{V}[u]\mathbb{E}[v^2]$. Independence alone does not justify replacing the variance of a product by the product of variances.

Weight-gradient variance

For tanh near zero, $\sigma' \approx 1$, so the component gradient in (6) becomes

$$\frac{\partial L}{\partial W_k[i,j]} \approx \frac{\partial L}{\partial \mathbf{h}_k[i]}\mathbf{h}_{k-1}[j].$$

For the output layer, replace $\mathbf{h}_k$ by $\mathbf{y}$. Also neglecting dependence between the backward gradient and the forward activation, and using the centering assumptions,

$$\mathbb{V}\left[\frac{\partial L}{\partial W_k[i,j]}\right] \approx \mathbb{V}\left[\frac{\partial L}{\partial \mathbf{h}_k[i]}\right]\mathbb{V}[\mathbf{h}_{k-1}[j]]. \tag{14}$$

The first factor propagates backward from the output and the second forward from the input. Expanding them using the main-text recurrences gives

$$\begin{aligned}\mathbb{V}\left[\frac{\partial L}{\partial W_k[i,j]}\right] \approx \mathbb{V}\left[\frac{\partial L}{\partial \mathbf{y}[i]}\right]\mathbb{V}[\mathbf{x}[j]] \\ \times \left(\prod_{t=k+1}^{M+1} N_t\mathbb{V}[W_t[i,j]]\right)\left(\prod_{t=1}^{k-1} N_{t-1}\mathbb{V}[W_t[i,j]]\right).\end{aligned}$$

Empty products equal one. If the *fan-out* choice $\mathbb{V}[W_t[i,j]] = 1/N_t$ is used at every layer, the first product is one and the second telescopes:

$$\prod_{t=1}^{k-1}\frac{N_{t-1}}{N_t} = \frac{N_0}{N_{k-1}}.$$

Consequently,

$$\mathbb{V}\left[\frac{\partial L}{\partial W_k[i,j]}\right] \approx \frac{N_0}{N_{k-1}}\mathbb{V}\left[\frac{\partial L}{\partial \mathbf{y}[i]}\right]\mathbb{V}[\mathbf{x}[j]].$$

For $k = 2$, this is the slides' N_0/N_1 expression. It is a fan-out special case, not generally the result of the Xavier compromise. Even here, different widths can give different weight-gradient variances.

B Self-Check Answers

1. $\mathbf{J}_g \in \mathbb{R}^{3\times5}$ and $\partial L/\partial \mathbf{x} \in \mathbb{R}^{5\times1}$. The product $\mathbf{J}_g^\top(\partial L/\partial \mathbf{y})$ has shape $(5\times3)(3\times1)$.
2. $\boldsymbol{\delta}_m \in \mathbb{R}^{4\times1}$, $\nabla_{W_m} L \in \mathbb{R}^{4\times6}$, and $\nabla_{\mathbf{b}_m} L \in \mathbb{R}^{4\times1}$. The weight gradient is the outer product of a 4×1 local gradient and a 1×6 layer input.
3. No: differentiating the mean loss already includes $1/B$. For a value used by two branches, sum the two chain-rule contributions; neither contribution should overwrite the other.
4. Different random weights already distinguish hidden units, even with zero biases. Identical incoming and outgoing parameters preserve a hidden-unit permutation symmetry under deterministic gradient updates.
5. No. Exact preservation would require both $N_{k-1}\mathbb{V}[W_k[i,j]] = 1$ and $N_k\mathbb{V}[W_k[i,j]] = 1$, which imply $N_{k-1} = N_k$. Xavier uses a compromise when widths differ.
6. A uniform distribution on $[-a, a]$ has variance $a^2/3$. To obtain variance $2/(N_{k-1}+N_k)$, its squared bound must be $6/(N_{k-1}+N_k)$. A Gaussian's standard deviation is simply the square root of the target variance.

References

[1] H. Robbins and S. Monro. "A stochastic approximation method." *The Annals of Mathematical Statistics*, 22(3):400–407, 1951. doi:10.1214/aoms/1177729586.

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning representations by back-propagating errors." *Nature*, 323:533–536, 1986. doi:10.1038/323533a0.

[3] X. Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks." *AISTATS*, PMLR 9:249–256, 2010. Paper and PDF.

[4] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. "Automatic differentiation in machine learning: a survey." *JMLR*, 18(153):1–43, 2018. Paper and PDF.