

Pattern Recognition and Machine Learning



Lecture 17 Training Deep Networks

Chizhi Chris ZHANG

zhangchizhi@ciomp.ac.cn

Advanced Computing and Digital Technology Research Center

University of Chinese Academy of Sciences

Autumn 2026

Lecture plan and time allocation

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Topic	Minutes
Parameter updates	45
Training signals and data	45
Diagnosis and transfer	45

The state of a training run

Follow updates; distinguish training and evaluation behavior; diagnose from evidence.

The numerical steps are shown on the slides.



One training step has five linked operations

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Keep the old state, gradient history and new parameter separate at each update.

Let the optimizer combine g_t with its stored state and update θ .

$$L_B = \frac{1}{B} \sum_i \ell(\mathbf{f}_\theta(\mathbf{x}_i), \mathbf{y}_i) \quad \mathbf{g}_t = \nabla_\theta L_B$$

Check. Old gradients accumulate unless they are cleared intentionally.



SGD: calculate the gradient before the step

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Keep processed samples and training time visible when changing batch size.

$$\nabla L(\theta) = \frac{1}{N} \sum_{i=1}^N \nabla \ell_i(\theta), \quad g_B = \frac{1}{B} \sum_{i \in B} \nabla \ell_i(\theta).$$

Different batches produce different steps. Averaging loss rather than summing it keeps the learning-rate meaning less sensitive to batch size.



$$g_B = (2 - 1 + 3 + 0)/4 = 1.$$

$$\theta_t = 5 \quad \eta = 0.1$$

$$\theta_{t+1} = 5 - 0.1 \times 1 = 4.9$$

Selected batch	Mean gradient	Next parameter
Samples 1 and 2	0.5	4.95
Samples 3 and 4	1.5	4.85
All four samples	1.0	4.90

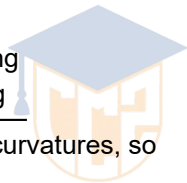
Learning rate controls how far, not which way

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

For $L(w) = \frac{1}{2}w^2$, gradient descent gives $w_{t+1} = (1 - \eta)w_t$. Start at $w_0 = 2$. The sign and magnitude of this factor determine shrinking, oscillation or growth.

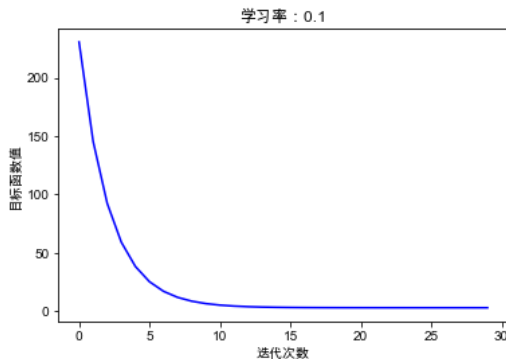
Learning rate	Successive parameter values	Behavior
0.1	2, 1.8, 1.62, ...	Slow contraction
1.5	2, -1, 0.5, ...	Alternating, shrinking
2.1	2, -2.2, 2.42, ...	Alternating, growing

Check. This quadratic converges for $0 < \eta < 2$. Neural networks have different curvatures, so this interval is not a universal learning-rate rule.



Check the loss after updates with one learning rate

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER



The horizontal axis counts updates and the vertical axis gives the objective. This run with learning rate .1 decreases the loss. A single curve cannot establish the best rate for a different task.



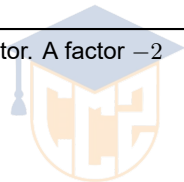
A narrow valley makes plain SGD zigzag

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Accumulate consistent directions with momentum, or rescale coordinates using gradient history.

Learning rate	Factor for coordinate 1	Factor for coordinate 2
.01	$1 - .01 = .99$	$1 - 100(.01) = 0$
.03	$1 - .03 = .97$	$1 - 100(.03) = -2$

Here $L = \frac{1}{2}(w_1^2 + 100w_2^2)$ and each coordinate update multiplies its current value by its factor. A factor -2 alternates the sign and doubles the magnitude.



Momentum: work through two different gradient histories

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

g_t : current mini-batch gradient; m_t : accumulated direction; $0 \leq \beta < 1$: memory strength.

$$m_0 = 0$$

$$m_t = \beta m_{t-1} + g_t, \quad \theta_{t+1} = \theta_t - \eta m_t.$$

$$0 \leq \beta < 1 \quad m_t = g_t + \beta g_{t-1} + \beta^2 g_{t-2} + \dots$$

$$v = -\eta m$$

Momentum can accelerate a stable direction and smooth an alternating one; it can also overshoot when an old direction becomes stale.

$$\theta_0 = 5 \quad \eta = 0.1$$

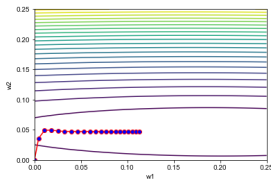
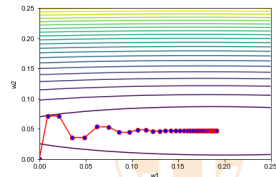
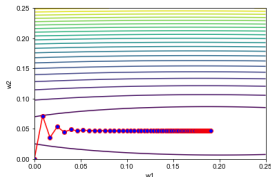
$$\beta = 0.9 \quad m_0 = 0$$

History	m_1, θ_1	m_2	θ_2
$g_1 = 2, g_2 = 2$	2, 4.8	$0.9(2) + 2 = 3.8$	4.42
$g_1 = 2, g_2 = -2$	2, 4.8	$0.9(2) - 2 = -0.2$	4.82

Read the optimization paths in the reference figure

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Plain descent repeatedly crosses the steep direction. Momentum smooths crossings while advancing along the valley.



Squared-gradient history rescales each coordinate

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

AdaGrad: accumulate squared gradients

$$\mathbf{s}_t = \mathbf{s}_{t-1} + \mathbf{g}_t^2.$$

The history grows and does not forget. Large accumulated values reduce later steps.

$$\theta_{t+1} = \theta_t - \eta \frac{\mathbf{g}_t}{\sqrt{\mathbf{s}_t} + \epsilon}.$$

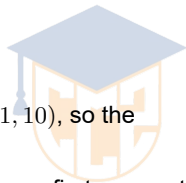
All operations are coordinatewise. With AdaGrad's initial $\mathbf{s}_0 = 0$ and $\mathbf{g}_1 = (1, 10)$, $\sqrt{\mathbf{s}_1} = (1, 10)$, so the normalized direction is approximately $(1, 1)$.

Squared history controls scale, not sign. The gradient supplies direction. Adam will combine a first-moment direction with a second-moment scale.

RMSProp: average squared gradients

$$\mathbf{s}_t = \rho \mathbf{s}_{t-1} + (1 - \rho) \mathbf{g}_t^2.$$

The moving average reduces the influence of distant history.



Adam combines direction, scale and bias correction

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Key idea

Coordinate adaptation changes effective steps. It does not remove the need for a suitable global η .

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \mathbf{g}_t,$$

$$\hat{\mathbf{m}}_t = \mathbf{m}_t / (1 - \beta_1^t),$$

$$\theta_{t+1} = \theta_t - \eta \hat{\mathbf{m}}_t / (\sqrt{\hat{\mathbf{v}}_t} + \epsilon).$$

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) \mathbf{g}_t^2,$$

$$\hat{\mathbf{v}}_t = \mathbf{v}_t / (1 - \beta_2^t),$$

$$\beta_1 = 0.9, \beta_2 = 0.999$$

Earlier momentum used $\mathbf{m}_t = \beta \mathbf{m}_{t-1} + \mathbf{g}_t$. Adam uses $(1 - \beta_1) \mathbf{g}_t$ in an exponential average. These are different normalization conventions.



Adam: the first update is easy to check

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Start Adam with $\mathbf{g}_1 = 2$, $\mathbf{m}_0 = \mathbf{v}_0 = 0$, $\beta_1 = .9$, $\beta_2 = .999$ and $\eta = .1$. Bias correction compensates for the zero-initialized moving averages before the first parameter update.

Quantity	Substitute	Result
$\mathbf{m}_1$	$0.1(2)$	0.2
$\mathbf{v}_1$	$0.001(4)$	0.004
$\hat{\mathbf{m}}_1$	$0.2/(1 - 0.9)$	2
$\hat{\mathbf{v}}_1$	$0.004/(1 - 0.999)$	4
update	$-0.1(2)/(\sqrt{4} + \epsilon)$	approximately -0.1



Check. Later updates depend on the full gradient history, and a near-zero denominator needs ϵ .

Compare two schedules over the same training budget

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

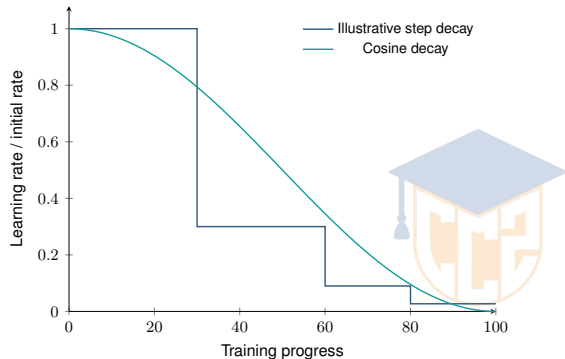
A large early rate can make rapid progress; a smaller late rate can refine the solution.

Both schedules share the same training budget. Step decay changes the rate at specified milestones; cosine decay changes it smoothly. Compare when each permits large or small updates, then validate the choice.

$$\eta_t = \frac{\eta_0}{2} [1 + \cos(\pi t/T)] \quad \eta_0 = 0.01$$

$$t = 0, T/2, T$$

$\eta_0 = 0.01$, $t = 0, T/2, T$ giving respectively 0.01, 0.005, 0.



Warmup and optimizer choice need a controlled comparison

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Start with a small rate and increase to the intended rate over the first W updates: $\eta_t = \eta_{\max} t / W$ for $t \leq W$. Then apply the main schedule.

$$\eta_t = \eta_{\max} t / W \quad W = 1000$$

$$\eta_{\max} = 10^{-3}$$

$W = 1000, \eta_{\max} = 10^{-3}$, $t = 1, 100, 1000$ gives learning rates $10^{-6}, 10^{-4}, 10^{-3}$; apply the main schedule after warmup.

Warmup update t	1	100	1000
Rate, for $W = 1000$ and $\eta_{\max} = 10^{-3}$	10^{-6}	10^{-4}	10^{-3}

To resume the same training path, save parameters, optimizer histories, the update count and scheduler progress.

Xavier and He scaling follow the layer size

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Updates can fail because of activation scale, changing statistics or invalid training transformations.

Keep forward activations and backward gradients at a usable scale rather than choose arbitrary small random numbers.

The argument uses distributional assumptions. Normalization and trained weights change these distributions; initialization is a starting design.

$$z_j = \sum_{i=1}^{n_{\text{in}}} w_{ji} x_i$$

$$\text{Var}(z_j) \approx n_{\text{in}} \text{Var}(w) \text{Var}(x).$$

$$n_{\text{in}} = 100$$

A common ReLU initialization uses $\text{Var}(w) = 2/n_{\text{in}}$. $n_{\text{in}} = 100$ gives variance .02 and standard deviation $\sqrt{0.02} \approx 0.1414$.



BatchNorm normalizes a feature across a mini-batch

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Input standardization uses statistics fitted on the training set and then applies the same transformation to new data.

Key idea

BatchNorm compares the same feature across examples; it does not normalize all numbers in a batch indiscriminately.

$$\mu_B = \frac{1}{B} \sum_i z_i, \quad \sigma_B^2 = \frac{1}{B} \sum_i (z_i - \mu_B)^2, \quad y_i = \gamma \frac{z_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta.$$



BatchNorm: calculate normalized and learned outputs

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

The normalized batch has mean zero and variance approximately one.

Worked calculation

$$\mathbf{z} = (1, 3, 5, 7)$$

$$\mu_B = 4, \quad \sigma_B^2 = (9 + 1 + 1 + 9)/4 = 5.$$

Input $\mathbf{z}_i$	1	3	5	7
After normalization, $(\mathbf{z}_i - 4)/\sqrt{5}$	-1.342	-0.447	0.447	1.342
; then $\gamma = 2, \beta = 1$	-1.683	0.106	1.894	3.683

BatchNorm has different training and evaluation behavior

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

A single test image should not need an artificial test batch to define its normalization.

Check. If new-camera activations differ strongly from training, saved running statistics can become inappropriate.

Setting	BatchNorm statistics	Does it control gradient recording?
Training mode	Current mini-batch; update running statistics	No: mode and recording are separate choices.
Evaluation mode	Saved running mean and variance	No: use the needed recording context separately.
No-gradient context alone	Does not switch the layer to evaluation behavior	It disables gradient recording.

LayerNorm normalizes features within one example

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Two rows are two examples; columns are two features. Use $\gamma = 1$, $\beta = 0$ and omit tiny ε only for arithmetic.

$$Z = \begin{bmatrix} 1 & 3 \\ 5 & 7 \end{bmatrix}.$$

Compare down each column

$$\mu = (3, 5), \quad \mathbf{s} = (2, 2).$$

$$\hat{Z}_{\text{column}} = \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix}.$$

Each feature compares values from different examples, as in this BatchNorm arithmetic.

The original value 3 becomes -1 by columns and $+1$ by rows. Specify the normalized axes before interpreting a value.

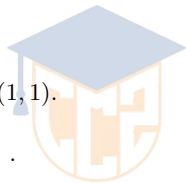
For one example with D features: $\mu = D^{-1} \sum_j h_j$, $\sigma^2 = D^{-1} \sum_j (h_j - \mu)^2$, and $y_j = \gamma_j (h_j - \mu) / \sqrt{\sigma^2 + \varepsilon} + \beta_j$.

Compare across each row

$$\mu = (2, 6), \quad \mathbf{s} = (1, 1).$$

$$\hat{Z}_{\text{row}} = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}.$$

Each example compares its own features, as in this LayerNorm arithmetic.



A residual path gives a layer an identity option

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

A block learns a correction F to the input rather than the entire mapping from scratch.

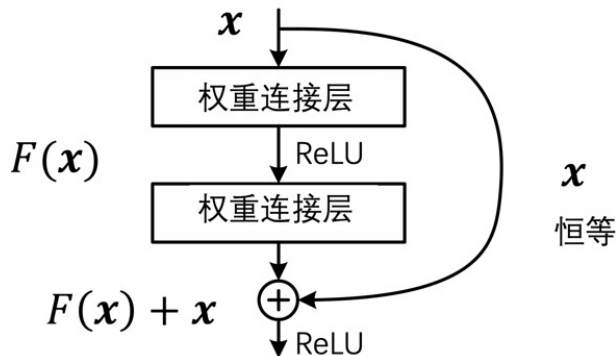
$$h_{\ell+1} = h_{\ell} + F_{\ell}(h_{\ell}), \quad \frac{\partial h_{\ell+1}}{\partial h_{\ell}} = I + \frac{\partial F_{\ell}}{\partial h_{\ell}}.$$

$$h = 3 \quad F(h) = -0.2 \Rightarrow h_{\text{new}} = 3 - 0.2 = 2.8$$



Add the learned correction to a shortcut

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER



One path computes $F(x)$; the shortcut carries x . The block adds matching shapes, then applies ReLU in this diagram. The shortcut supplies an identity path before that final activation.



Regularization methods act at different locations

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Where is the problem: fitting the training set, or generalizing beyond it?

Method	Change what?	Intended effect
Weight decay	Parameter Update	Limit parameter scale
Dropout	Training activations	Reduce reliance on fixed feature combinations
Data augmentation	Training examples supplied	Learn invariance to permitted transformations
Early stopping	Retained checkpoint	Stop fitting further training details

Check. If both training and validation are poor, check optimization, inputs, labels and capacity before adding stronger restrictions.

AdamW decouples shrinkage from adaptive gradients

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

SGD with an L2 gradient

combines shrinkage and the data gradient:

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

In an adaptive optimizer, adding λw inside the gradient also changes its adaptive history and scaling.

AdamW applies weight shrinkage separately from the normalized data direction u_t :

$$u_t = \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \varepsilon}}, \quad w_{t+1} = (1 - \eta\lambda)w_t - \eta u_t.$$



One scalar update

Take $w = 2$, normalized direction $u_t = .5$, $\eta = .1$, $\lambda = .01$.

$$w_{t+1} = .999(2) - .1(.5) = 1.948.$$

Without decay the update gives 1.95. Here .5 is the normalized direction, not necessarily the raw gradient.

Dropout: one mask is noisy, its expectation is not

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

For drop probability p and independent mask $r_j \sim \text{Bernoulli}(1 - p)$, inverted dropout uses

$$\tilde{h}_j = \frac{r_j h_j}{1 - p} \quad \text{during training ,} \quad \tilde{h}_j = h_j$$

during evaluation .

$$\mathbb{E}[\tilde{h}_j] = h_j$$

A nonlinear network's full prediction is not exactly the average prediction of every masked subnetwork. The ensemble analogy is useful but approximate.

Worked calculation

$$h = (2, 4) \quad p = 0.5$$

Mask r	Training Output $\tilde{h}$	Probability
(0, 0)	(0, 0)	1/4
(1, 0)	(4, 0)	1/4
(0, 1)	(0, 8)	1/4
(1, 1)	(4, 8)	1/4

$$\mathbb{E}[\tilde{h}] = \frac{(0, 0) + (4, 0) + (0, 8) + (4, 8)}{4} = (2, 4) = h.$$

Training mode is a layer behavior, not a gradient switch

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

An unchanged weight tensor does not necessarily imply unchanged model outputs.

Switch	Change what?
Training or evaluation mode	BatchNorm statistics and Dropout randomness
Gradient recording	Whether operations are recorded for differentiation
Parameter freezing	Whether parameters receive gradients and optimizer updates

Condition to check

BatchNorm running statistics can change in training mode even when weights do not update.

An augmentation needs a valid target and a separate evaluation

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

A translation can preserve a defect label if the crack remains visible. A crop can remove the only crack while retaining a positive target, producing incorrect supervision.

Stage	Operation	Label or evidence rule
Before augmentation	Split the original cases	All variants of one case stay in one split.
Training	Apply a permitted random transform	Preserve class evidence or update spatial targets.
Evaluation	Apply fixed preprocessing	Compare models under the same input rules.

Restrict the crop, update a spatial label, or exclude an invalid transformation.

Read training and validation curves together

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

A lower training loss alone cannot tell us how the model behaves on new objects.

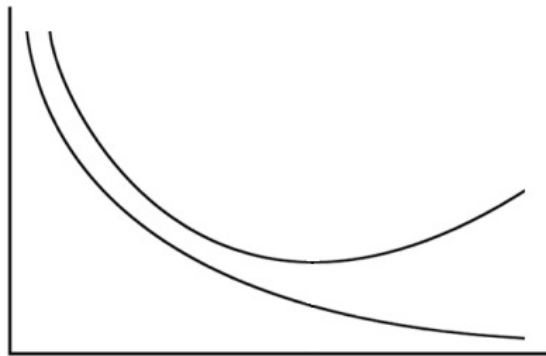
It narrows the possibilities. It does not uniquely identify the cause from shape alone.

Observation	First check
Both losses high	Tiny-set fitting; input, target, gradient and loss checks
Low training loss; high validation loss	Split and distribution differences; data, regularization and capacity
Both losses still decreasing	Training budget, learning rate and continuing improvement
Sudden NaN	Locate the first invalid tensor; check scale and numerical range

First fit a small verified dataset successfully, then investigate generalization on the full task.

Find the best validation checkpoint

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER



Horizontal: training progress. The falling curve is training error; the U-shaped curve is validation error.



A learning-curve example leads to a concrete decision

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

The model keeps fitting training cases, but validation loss worsens after epoch 10.

Round	1	5	10	20
Training cross-entropy	1.10	0.55	0.22	0.08
Validation cross-entropy	1.15	0.64	0.61	0.82
Validation accuracy	55%	73%	76%	75%

Check. Do not choose the epoch or intervention using the final test set.



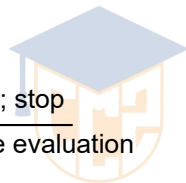
Early stopping selects a checkpoint, not a magic epoch

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

The selected training horizon limits how far the model fits idiosyncrasies of the training set.

Round	Validation cross-entropy	Processing
10	0.61	Save current best checkpoint
11	0.63	One epoch without improvement
12	0.65	Two epochs without improvement
13	0.66	Three epochs without improvement; stop

Restore the best saved checkpoint at epoch 10. Compare losses using the same evaluation mode and preprocessing.



Fit a tiny subset before changing the architecture

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Pick 16–32 correctly labeled training examples and inspect them manually.

Condition to check

A successful tiny-subset fit checks that learning is possible. It does not demonstrate generalization or justify the final architecture.

Tiny verified training subset	First interpretation	Next check
Cannot be fitted	The learning chain may be broken	Inspect inputs, labels, output/loss compatibility, gradients and rate.
Can be fitted	The model can learn these training cases	Return to independent validation to assess generalization.
Good tiny-subset accuracy	A deliberate debugging result	Do not report it as an independent test score.

Transfer example: count parameters and choose a first test

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Let a pretrained backbone produce features $h_\phi(x)$ and a new head predict

$$p(y | x) = \text{softmax}(Wh_\phi(x) + b).$$

Match the pretrained model's input format, scale and channel convention.

Check. A large uniform rate can overwrite representations before the new head becomes useful.

Begin with the frozen-head baseline, then compare selective tuning under the same split. Parameter count alone does not prove which option wins.

$$2 \times 512 + 2 = 1026 \quad \text{trainable parameters} .$$

Data size and domain difference guide how much to tune

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Compare against frozen features on target validation data. The table offers a starting point, not a performance guarantee.

Label count	Domain or task difference	First comparison
Less	Similar	Freeze backbone; train the new head
More	Similar	Train the new head, then gradually unfreeze upper layers
Less	Large	Check input and task compatibility; keep a simple baseline
More	Large	Compare broader tuning and suitable alternative starting models

Record trainable parameters, validation performance and training budget. Label count alone does not determine the best tuning strategy.

A minimal training loop puts the ideas into code

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

```
model.train()
for x, y in train_loader:
    optimizer.zero_grad(set_to_none=True)
    logits = model(x)
    loss = criterion(logits, y)
    loss.backward()
    torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
    optimizer.step()
# scheduler.step() goes at its documented update/epoch boundary
```

Clear previous gradients;
compute the current loss and
gradients; then update.



Validation measures the model without training it

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

```
model.eval()
total_loss, correct, count = 0.0, 0, 0
with torch.no_grad():
    for x, y in valid_loader:
        logits = model(x)
        loss = criterion(logits, y) # mean over this batch
        total_loss += loss.item() * y.size(0)
        correct += (logits.argmax(1) == y).sum().item()
        count += y.size(0)
valid_loss, valid_acc = total_loss / count, correct / count
```

The final batch can be smaller.
Weight batch means by case
count to obtain the correct
dataset mean. No backward or
optimizer step belongs here.



Different training evidence calls for different checks

ADVANCED COMPUTING AND DIGITAL TECHNOLOGY RESEARCH CENTER

Case A: 99% training, 73% validation

Check whether device groups and recording conditions match the future task. Inspect RGB order, pixel range and defect visibility.

Correct an input mismatch first, then compare regularization or selective tuning under the same split.

Case B: both losses are high

Inspect inputs and targets. Fit a tiny verified subset and check loss and gradients.

If this works but validation remains poor, investigate split mismatch, data volume and regularization.

Change one mechanism at a time; curve shape alone does not identify the cause.

