Generalization and Regularization

Diagnose the gap, change one lever, measure again

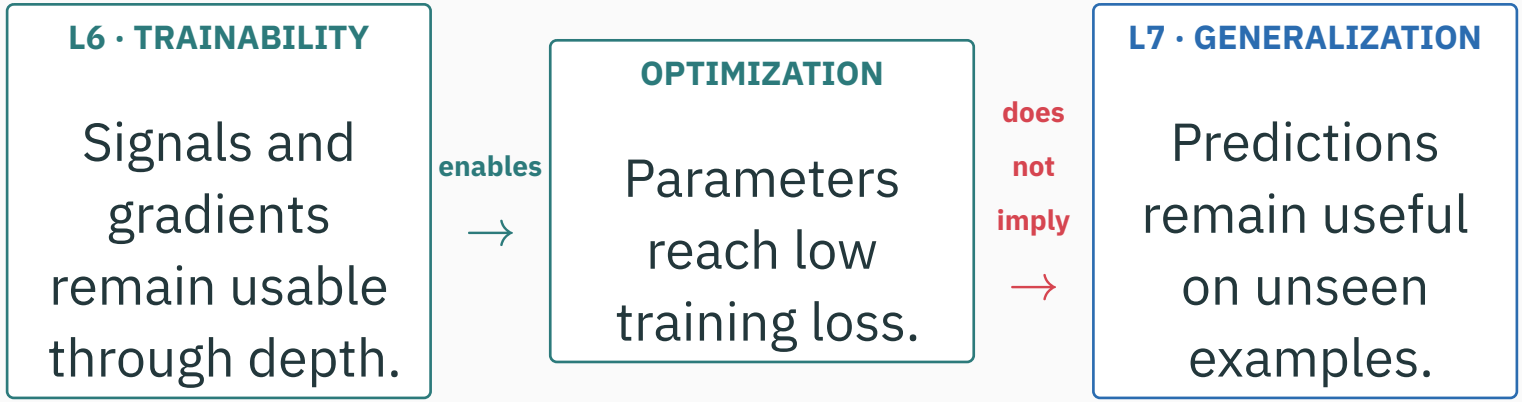
Prof. Nipun Batra

ES 667 · Deep Learning · IIT Gandhinagar

Diagnose before prescribing

L6 made the network trainable; L7 asks what survives new data

V VISUAL



Today's unit of reasoning is not a named trick: it is symptom → suspect → controlled test.

Commit: what should we do with 99% train and 72% validation?

Q QUESTION

RUNNING CASE · FIXED TEACHING TRACE · five-class image classifier · chance 20% · train 99% · validation 72% on 100 examples · gap 27 points · test sealed

Choose the first defensible action. Keep your answer until the closing revisit.

A. Deploy: 99% training accuracy shows the classifier is ready

B. Add every regularizer at once and keep the best test score

C. Inspect validation curves, preserve the test seal, then change one lever

D. Tune directly on the test set because validation is lower

Write **A, B, C,** or **D** plus one sentence of evidence. No answer is revealed yet.

Keep the commitment; every section updates one ledger

RUNNING CASE · FIXED TEACHING TRACE · five-class image classifier · chance 20% · train 99% · validation 72% on 100 examples · gap 27 points · test sealed

CASE LEDGER · OBSERVE → INTERVENE → MEASURE

gap / split	observed · $99 - 72 = 27$ points; cause not yet identified
checkpoint	validation curve not yet inspected; test remains sealed
weights	pending
representations	pending
data	pending
targets	pending

The ledger separates **evidence already measured** from **the next intervention to test**.

Outline: diagnose, intervene, verify

V VISUAL

> Every method enters at the third beat; the evidence protocol stays unchanged.

training validation intervention accepted evidence failure sealed test

A gap is a diagnostic, not a verdict

D DERIVATION

RUNNING CASE · FIXED TEACHING TRACE · five-class image classifier · chance 20% · train 99% · validation 72% on 100 examples · gap 27 points · test sealed

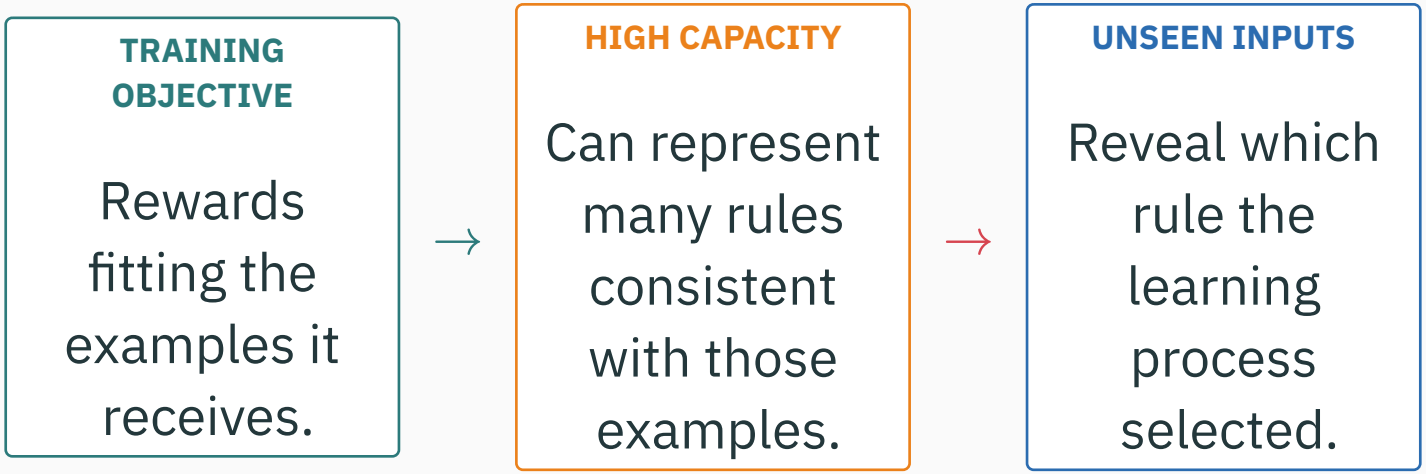
$$\text{gap} = \text{Acc}_{\text{train}} - \text{Acc}_{\text{val}} = 0.99 - 0.72 = 0.27.$$

WHAT IT SAYS	WHAT IT DOES NOT SAY	NEXT EVIDENCE
performance differs by 27 percentage points on this fixed split	which remedy will help, or whether 72% meets the deployment target	training and validation loss by epoch, plus split integrity

Regularize only after the measurements distinguish underfitting from overfitting.

Low training loss can coexist with memorization

V VISUAL



Deep networks can fit random labels, so training fit alone does not identify a useful rule. Evidence: Zhang et al. (2017), **Understanding deep learning requires rethinking generalization**.

This claim motivates a measurement protocol; it does not claim that our five-class labels are random.

Split the decisions, then stop on validation

Three splits have three different jobs

V VISUAL

split	may affect	must not affect
train	weights	final reported metric
validation	model and checkpoint choices	gradient updates in the basic protocol
test	final report only	weights, hyperparameters, or retry decisions

Why does repeated test checking leak information?

Q QUESTION

After every run, a team reports test accuracy and chooses the next λ. What has the test set become?

A. A larger training set

B. A validation set used indirectly for selection

C. An unbiased final estimate with lower variance

D. A regularizer because the labels were not differentiated

> Commit before the next slide. Name the decision that consumed test information.

Answer: selection turns test into validation

A ANSWER

Correct: B — a validation set used indirectly for selection

Why: The chosen λ depends on test outcomes. The final reported score is therefore conditional on having searched that same set.

Seal the test set; if it changes a decision, it was not a final test.

Which checkpoint should survive?

Q QUESTION

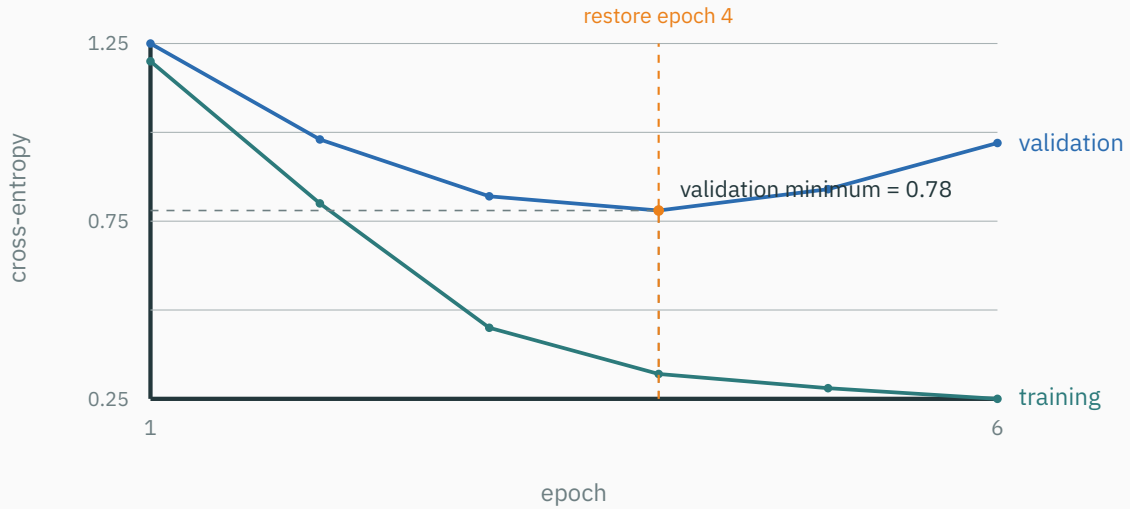
RUNNING CASE · FIXED TEACHING TRACE · five-class image classifier · chance 20% · train 99% · validation 72% on 100 examples · gap 27 points · test sealed

epoch	1	2	3	4	5	6
train loss	1.20	0.80	0.45	0.32	0.28	0.25
validation loss	1.25	0.98	0.82	0.78	0.84	0.97

Choose the checkpoint and explain why epoch 6 is not automatically best.

Restore epoch 4: it is the observed validation minimum

A ANSWER



SELECTION

minimum validation loss is 0.78 at epoch 4

PATIENCE = 2

epochs 5 and 6 fail to improve; stop after epoch 6, restore epoch 4

The plotted points are exactly the six values in the preceding table; both curves share one loss axis.

Ledger update: the first remedy is a checkpoint, not a new model

RUNNING CASE · FIXED TEACHING TRACE · five-class image classifier · chance 20% · train 99% · validation 72% on 100 examples · gap 27 points · test sealed

CASE LEDGER · OBSERVE → INTERVENE → MEASURE

gap / split	observed · 27 points on the fixed validation split; test sealed
checkpoint	keep · epoch 4, validation loss 0.78; patience 2 stops after epoch 6
weights	pending
representations	pending
data	pending
targets	pending

Early stopping limits how far optimization follows the training objective. It does not repair a broken split or distribution shift.

Constrain the weights

Weight decay changes which fitting solution is preferred

V VISUAL

DATA FIT

$\mathcal{L}_{\text{train}(\theta)}$ rewards agreement with training labels

SIZE PREFERENCE

$\frac{\lambda}{2}\|\theta\|_2^2$ penalizes large parameter coordinates

$$\mathcal{J}(\theta) = \mathcal{L}_{\text{train}(\theta)} + \frac{\lambda}{2}\|\theta\|_2^2.$$

> The factor $\frac{1}{2}$ is a convention: it makes $\nabla_\theta\left(\lambda\frac{\|\theta\|^2}{2}\right) = \lambda\theta$. Always read the implementation before comparing numerical λ values.

Derive the shrinkage step before naming the optimizer

D DERIVATION

Let $g_t = \nabla_\theta \mathcal{L}_{\text{train}(\theta_t)}$. For plain SGD on the objective above,

$$\theta_{t+1} = \theta_t - \eta(g_t + \lambda\theta_t).$$

$$\theta_{t+1} = \underbrace{(1 - \eta\lambda)\theta_t}_{\text{shrink}} \quad \underbrace{- \eta g_t}_{\text{fit data}}.$$

INTERVENTION

λ controls multiplicative shrinkage

MEASUREMENT

validation loss chooses λ; weight norm is a mechanism check

Calculate one scalar update completely

D DERIVATION

RUNNING CASE · FIXED TEACHING TRACE · five-class image classifier · chance 20% · train 99% · validation 72% on 100 examples · gap 27 points · test sealed

Given $\theta_t = 5$, training gradient $g_t = 3$, learning rate $\eta = 0.1$, and decay rate $\lambda_{\text{wd}} = 0.2$:

$$1 - \eta\lambda_{\text{wd}} = 1 - (0.1)(0.2) = 0.98.$$

$$\theta_{t+1} = \underbrace{0.98 \times 5}_{\text{decay}} - \underbrace{0.1 \times 3}_{\text{gradient}} = 4.90 - 0.30 = 4.60.$$

Decay accounts for 0.10 of the change; the data gradient accounts for 0.30.

AdamW keeps decay outside the adaptive gradient transform

V VISUAL

COUPLED L2

$\theta^+ = \theta - \eta P_{t(g+\lambda\theta)}$ The penalty enters adaptive moments and preconditioning.

DECOUPLED ADAMW

$\theta^+ = (1 - \eta\lambda_{\mathrm{wd}})\theta - \eta P_{t(g)}$ Shrinkage is applied directly to the parameter.

These coincide for plain SGD up to convention, but not generally for adaptive optimizers. Primary source: Loshchilov & Hutter, **Decoupled Weight Decay Regularization**.

Honest evidence: norm is a mechanism; validation chooses the setting

V VISUAL

λ_{wd}	validation loss	$\|\theta\|_2$	selection
0	0.596	4.921	reject
0.03	0.456	2.195	best validation
0.20	0.482	0.928	reject: $0.482 > 0.456$

Recorded companion ring experiment in notebook 2; it is not fabricated evidence from the five-class anchor.

A smaller norm confirms shrinkage occurred; it does not by itself prove better generalization.

Ledger update: test one weight intervention at a time

RUNNING CASE · FIXED TEACHING TRACE · five-class image classifier · chance 20% · train 99% · validation 72% on 100 examples · gap 27 points · test sealed

CASE LEDGER · OBSERVE → INTERVENE → MEASURE

gap / split	observed · 27 points; test sealed
checkpoint	keep · epoch 4, validation loss 0.78
weights	candidate · decoupled λ_{wd}; scalar audit $5 \rightarrow 4.60$; select using validation
representations	pending
data	pending
targets	pending

Do not combine a new decay rate, dropout probability, and augmentation policy in the same comparison.

Perturb the representation

Dropout trains a sampled representation, not a smaller stored network

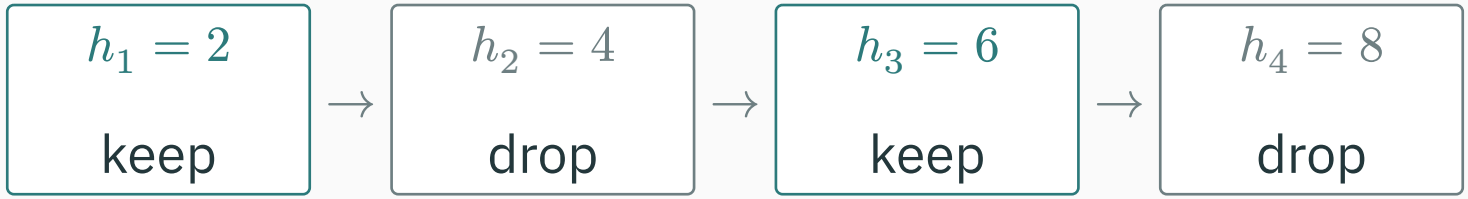
V VISUAL

ONE TRAINING PASS

Mask $m = (1, 0, 1, 0)$ keeps two hidden coordinates. The next layer sees a sampled feature set.

EVALUATION

The mask is disabled. All learned coordinates contribute through the complete network.



The rectangular layout shows values and mask state directly; no node or edge implies an unshown computation.

Inverted dropout preserves each activation in expectation

D DERIVATION

$$m_i \sim \text{Bernoulli}(p_{\text{keep}}), \quad \tilde{h}_i = m_i \frac{h_i}{p_{\text{keep}}}.$$

$$\mathbb{E}[\tilde{h}_i] = \mathbb{E}[m_i] \frac{h_i}{p_{\text{keep}}} = p_{\text{keep}} \frac{h_i}{p_{\text{keep}}} = h_i.$$

For $h = [2, 4, 6, 8]$, $m = [1, 0, 1, 0]$, and $p_{\text{keep}} = 0.5$:

$$\tilde{h} = \frac{[1 \times 2, 0 \times 4, 1 \times 6, 0 \times 8]}{0.5} = [4, 0, 12, 0].$$

> One mask is not equal to the original vector; the equality is an expectation over masks.

Name the probability: PyTorch takes the drop probability

V VISUAL

symbol / API	meaning	separate API example
p_{keep}	probability that a coordinate survives	0.8
$p_{\text{drop}} = 1 - p_{\text{keep}}$	probability that it is zeroed	0.2
`nn.Dropout(p)`	`p` means p_{drop}, not p_{keep}	`Dropout(p=0.2)`

```
import torch
h = torch.tensor([2., 4., 6., 8.])
drop = torch.nn.Dropout(p=0.2)  # p_drop = 0.2, p_keep = 0.8
drop.train(); y_train = drop(h)  # random, scaled by 1/(1-p)
drop.eval();  y_eval  = drop(h)  # identity
```

The preceding mask calculation used $p_{\text{keep}} = 0.5$; unequal values here expose the API conversion.

Training and evaluation must disagree in exactly one controlled way

V VISUAL

mode	dropout	what remains fixed
`model.train()`	random masks + inverted scaling	learned weights, input, loss definition
`model.eval()`	identity mapping	same learned weights; no manual rescaling

EXPECTED EFFECT

reduced reliance on any one hidden coordinate

MEASURE

validation loss and calibration; repeat seeds if stochastic variation matters

Mechanism and empirical effect are separate claims. Dropout's original analysis: Srivastava et al. (2014).

When is dropout the wrong first response?

Q QUESTION

Which observation argues against increasing dropout first?

A. Training loss is already high and validation loss is similarly high

B. Training loss is low while validation loss rises after epoch 4

C. The model relies on a few hidden features and validation is worse

D. A no-dropout baseline has lower train loss but higher validation loss

> Ask whether the model can fit before reducing its effective capacity.

Answer: high training loss is an underfitting signal

A ANSWER

Correct: A — training and validation losses are both high

Why: Stronger dropout makes fitting harder. First verify optimization, model capacity, features, and labels.

CASE LEDGER · OBSERVE → INTERVENE → MEASURE

gap / split	observed · 27 points; test sealed
checkpoint	keep · epoch 4
weights	candidate · decoupled decay; validate
representations	candidate · $p_{\text{keep}} = 0.5$ / PyTorch $p_{\text{drop}} = 0.5$; validate
data	pending
targets	pending

Regularize the examples and targets

Augmentation encodes a task-specific invariance

V VISUAL

For a transform T, the desired assumption is

$$y(T(x)) = y(x) \quad \text{for transformations allowed by the task.}$$

VALID FOR OUR CASE

A small translation or crop that keeps the object fully visible.

INVALID FOR OUR CASE

A crop that removes the class-defining object.

> The transform must preserve the label semantics, not merely look plausible to us.

Which augmentation policy is defensible?

Q QUESTION

RUNNING CASE · FIXED TEACHING TRACE · five-class image classifier · chance 20% · train 99% · validation 72% on 100 examples · gap 27 points · test sealed

For the five-class object classifier, which policy states a checkable label-preservation rule?

A. Apply any transform that increases training loss

B. Translate up to two pixels, rejecting samples where the object leaves the frame

C. Randomly replace the class label after every crop

D. Rotate by any angle without inspecting class semantics

Answer: constrain the transform by the task

A ANSWER

Correct: B — translate only while the object remains in frame

Why: The policy names both the transformation and the condition under which the label is preserved.

A transform is an intervention. Audit a sample grid and measure validation performance before keeping it.

Mixup trains behaviour between observed examples

V VISUAL

$$\tilde{x} = \alpha x_i + (1-\alpha)x_j, \quad \tilde{y} = \alpha y_i + (1-\alpha)y_j.$$

> Mixup's claim is about training on convex combinations, not arbitrary semantic invariance. Primary source: Zhang et al. (2018), **mixup: Beyond Empirical Risk Minimization**.

Calculate one five-class mixup target

D DERIVATION

Let $y_i = [1, 0, 0, 0, 0]$, $y_j = [0, 0, 0, 1, 0]$, and $\alpha = 0.25$.

$$\tilde{y} = 0.25 y_i + 0.75 y_j = [0.25, 0, 0, 0.75, 0].$$

For predicted class probabilities p,

$$\mathrm{CE}(\tilde{y}, p) = -0.25 \log p_1 - 0.75 \log p_4.$$

The target and loss use the same mixing coefficient as the input.

Label smoothing has two common conventions—name yours

V VISUAL

For $K = 5$ and $\varepsilon = 0.1$ with class 1 correct:

convention	correct class	each wrong class
other-class redistribution	$1 - \varepsilon = 0.90$	$\frac{\varepsilon}{K-1} = 0.025$
PyTorch uniform mixing	$1 - \varepsilon + \frac{\varepsilon}{K} = 0.92$	$\frac{\varepsilon}{K} = 0.02$

> PyTorch `CrossEntropyLoss(label_smoothing=ε)` mixes the one-hot target with a uniform distribution over **all** K classes: official API definition.

Write both smoothed targets; they are not interchangeable

D DERIVATION

OTHER-CLASS

$$[0.90, 0.025, 0.025, 0.025, 0.025]$$

PYTORCH UNIFORM

$$[0.92, 0.02, 0.02, 0.02, 0.02]$$

Both sum to 1, but they assign different mass to the correct class.

MECHANISM

discourages a zero-loss demand for probability exactly 1

MEASURE

validation NLL and calibration, alongside accuracy

Label smoothing was used in Szegedy et al. (2016), **Rethinking the Inception Architecture for Computer Vision**.

Accuracy and calibration answer different questions

V VISUAL

Use one validation confidence bin with 100 examples and 72 correct predictions.

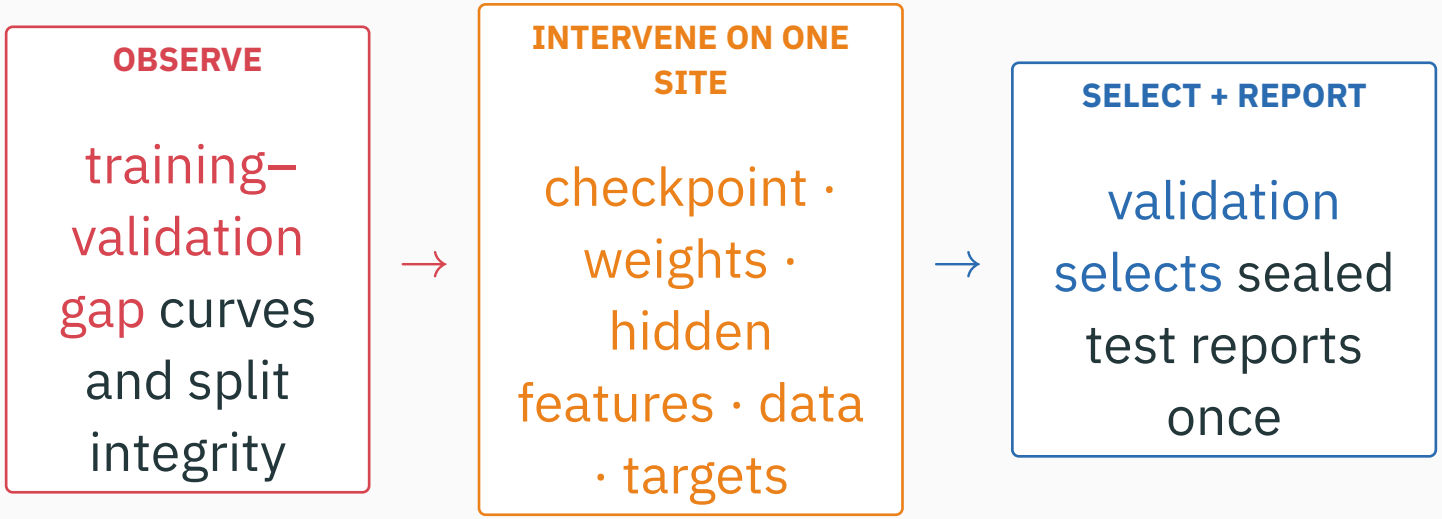
same predicted labels	version A	version B
accuracy	0.72	0.72
mean confidence	0.90	0.74
bin calibration gap	$\lvert 0.90 - 0.72 \rvert = 0.18$	$\lvert 0.74 - 0.72 \rvert = 0.02$

Same accuracy, different confidence quality. Report both when probabilities drive decisions.

Turn the diagnosis into one controlled experiment

One semantic map: observe → intervene → select → report

V VISUAL



TRAINING PATH + PARAMETERS	HIDDEN REPRESENTATION	EXAMPLES + TARGETS
early stopping selects a checkpoint; weight decay changes parameters	dropout perturbs hidden features during training	augmentation, mixup, and smoothing change supervision

Use symptom → suspect → test, not method → hope

V VISUAL

symptom	first suspect	controlled test
train and validation loss both high	underfit / optimization	repair fit before adding regularization
train falls; validation turns upward	overfit with epoch	early stop; restore validation minimum
low train; high validation across epochs	effective capacity / representation	sweep decay **or** dropout
accuracy stable; confidence too high	target pressure / calibration	declare smoothing convention; compare NLL

Pre-register the comparison before running it

D DERIVATION

RUNNING CASE · FIXED TEACHING TRACE · five-class image classifier · chance 20% · train 99% · validation 72% on 100 examples · gap 27 points · test sealed

baseline	same split, seed set, architecture, optimizer, and epoch budget
one change	e.g. $\lambda_{\text{wd}} \in \{0, 0.03, 0.20\}$; leave dropout and augmentation fixed
selection	minimum validation loss; restore that checkpoint
secondary measures	accuracy, NLL, calibration; norm only as a mechanism audit
final report	evaluate the sealed test once after the recipe is frozen

Two direct Colab labs, one prediction-first loop

I INTERACTIVE

1 · curves	01_overfitting_and_early_stopping.ipynb
2 · mechanisms	02_dropout_from_scratch.ipynb

PREDICT	RUN	EXPLAIN
write the expected curve or activation before execution	change exactly one declared lever	use validation evidence to keep or reject it

The two filenames above are the only two Colab links in this deck; both point directly to the public repository.

Final ledger: one case, six auditable decisions

RUNNING CASE · FIXED TEACHING TRACE · five-class image classifier · chance 20% · train 99% · validation 72% on 100 examples · gap 27 points · test sealed

CASE LEDGER · OBSERVE → INTERVENE → MEASURE

gap / split	observe · train 99%, validation 72%, gap 27 points; test sealed
checkpoint	keep · epoch 4 at validation loss 0.78
weights	test · decoupled decay; scalar $5 \to 4.60$; select by validation
representations	test · $h = [2, 4, 6, 8] \to [4, 0, 12, 0]$ for one mask; PyTorch $p_{\text{drop}} = 0.5$
data	test · translate/crop only while the object remains and label is preserved
targets	test · state smoothing convention; measure validation NLL and calibration

Close the loop

Retrieval: which sequence protects the claim?

Q QUESTION

Which workflow supports a defensible generalization claim?

A. Tune on train, select on test, report validation

B. Change several methods, keep the run with the best test accuracy

C. Diagnose curves, change one lever, select on validation, report test once

D. Choose the smallest parameter norm without checking predictions

> Commit before the answer. Identify where each split appears in your choice.

Answer: diagnosis precedes intervention

A ANSWER

Correct: C — diagnose curves, change one lever, select on validation, report test once

Why: The workflow aligns each dataset with one job and makes the effect of the intervention interpretable.

Training demonstrates fit; validation chooses; the sealed test estimates the frozen procedure.

Revisit the opening commitment

A ANSWER

choice	problem	verdict
A	training fit is not unseen-data evidence	reject
B	many changes are not attributable; test checking leaks	reject
C	curves → one lever → validation → sealed test	keep
D	test-driven tuning destroys the estimate	reject

The winning commitment was C: diagnose first, then run one controlled comparison.

Exit ticket: prescribe the smallest next test

New run: training loss falls to 0.18; validation loss reaches 0.44 at epoch 9, then rises to 0.61 by epoch 20.

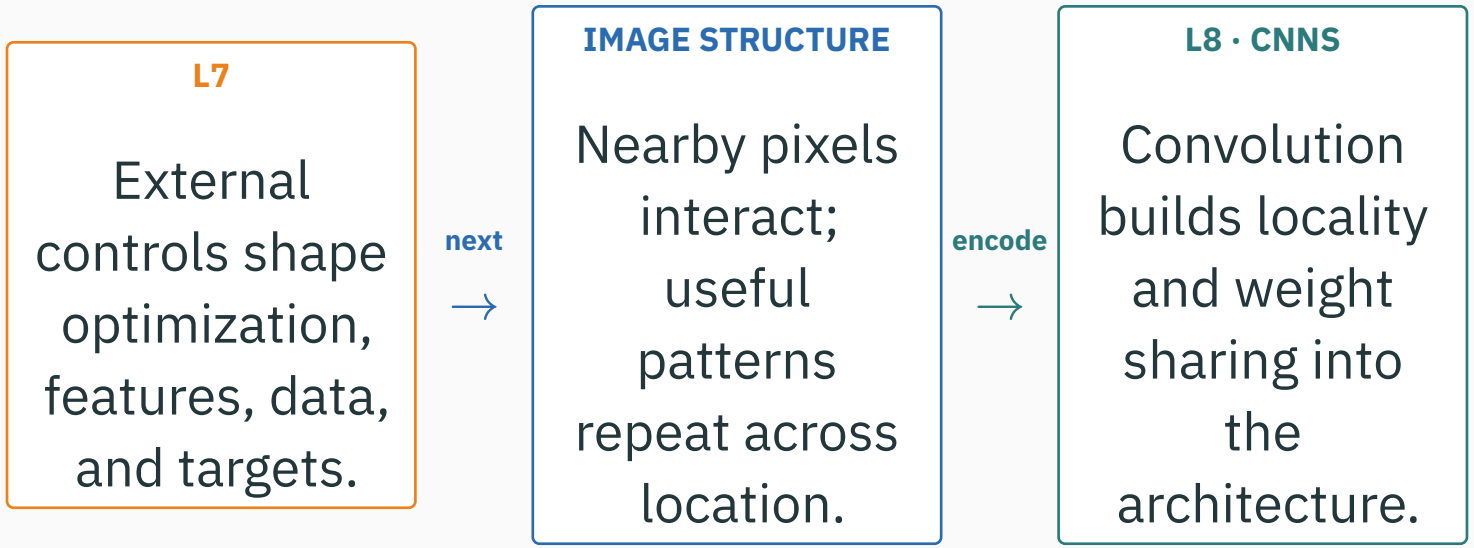
Write four lines:

1. **symptom:** what changed after epoch 9?
2. **suspect:** which mechanism is consistent with the curves?
3. **test:** what single intervention do you try first?
4. **measurement:** which checkpoint and metric decide?

> Strong first test: restore epoch 9, then compare one regularizer using validation—without consulting test.

L8 handoff: generalization can also come from architecture

V VISUAL



Regularization chooses among fitting rules; architectural inductive bias changes which rules are easy to express.

Diagnose the gap → change one lever → select on validation → report the sealed test once.

Next: convolution encodes locality and sharing as architectural inductive bias.

Implementation and provenance

Coupled penalty and decoupled decay: same for SGD, different for Adam

optimizer	coupled L2	decoupled decay
plain SGD	$\theta^{+} = (1 - \eta\lambda)\theta - \eta g$	same algebra after matching convention
adaptive method	$\theta^{+} = \theta - \eta P_{t(g+\lambda\theta)}$	$\theta^{+} = (1 - \eta\lambda_{\mathrm{wd}})\theta - \eta P_{t(g)}$

> The distinction is about where the penalty enters the update, not about whether norms tend to shrink.

Parameter groups are a recipe choice, not a theorem

```
decay, no_decay = [], []
for name, parameter in model.named_parameters():
    target = no_decay if (name.endswith("bias") or "norm" in name) else decay
    target.append(parameter)

optimizer = torch.optim.AdamW([
    {"params": decay, "weight_decay": 3e-2},
    {"params": no_decay, "weight_decay": 0.0},
], lr=1e-3)
```

> Excluding biases and normalization parameters is common, but architecture- and recipe-dependent. Log the groups you actually used.

Early stopping needs saved state, not only a stopping epoch

```
best, waits, snapshot = float("inf"), 0, None
for epoch in range(max_epochs):
    train_one_epoch(model)
    value = validation_loss(model)
    if value < best:
        best, waits = value, 0
        snapshot = copy.deepcopy(model.state_dict())
    else:
        waits += 1
    if waits >= patience:
        break
model.load_state_dict(snapshot)
```

> Save optimizer state too if training will resume. For final inference, restore the selected model state.

Calibration aggregates bin gaps; binning is part of the estimate

For bins B_m with accuracy $\mathrm{acc}(B_m)$ and mean confidence $\mathrm{conf}(B_m)$,

$$\mathrm{ECE} = \sum_m \left(\frac{|B_m|}{n} \right) |\mathrm{acc}(B_m) - \mathrm{conf}(B_m)|.$$

USE WITH CARE

ECE changes with bin boundaries and sample size

PAIR WITH

NLL, accuracy, reliability plot, and task costs

The earlier 0.18 and 0.02 values were single-bin gaps, not full-dataset ECE claims.

Companion notebook traces: concrete, separate from the anchor

source	setting	validation / agreement	mechanism / report
notebook 1	epoch 1640	MSE 0.1167 best; 0.1756 last	restored test MSE 0.1111
notebook 2	$\lambda_{\text{wd}} = 0$	loss 0.596	norm 4.921
	$\lambda_{\text{wd}} = 0.03$	loss 0.456	norm 2.195
	$\lambda_{\text{wd}} = 0.20$	loss 0.482	norm 0.928
transform audit	rotation	label agreement 1.000	data-specific
	translation	label agreement 0.838	data-specific

> These recorded outputs motivate checks; they are not substituted for the five-class trace, and model agreement alone is not proof of label preservation.

Primary references and vector provenance

- Zhang et al. (2017), Understanding deep learning requires rethinking generalization — memorization and random labels.
- Loshchilov & Hutter (2019), Decoupled Weight Decay Regularization — AdamW.
- Srivastava et al. (2014), Dropout: A Simple Way to Prevent Neural Networks from Overfitting.
- Zhang et al. (2018), mixup: Beyond Empirical Risk Minimization.
- Szegedy et al. (2016), Rethinking the Inception Architecture for Computer Vision — label smoothing.
- PyTorch, CrossEntropyLoss — uniform-mixture `label_smoothing` convention.

All diagrams, tables, and plotted traces in this deck are native Typst vectors computed from the values shown; no raster image is embedded. Double descent is deliberately omitted rather than assigned an unsupported threshold.