THE COST OF A RULE: WHY A BACKDOOR NEEDS A COUNT AND A WRONG LABEL NEEDS A FRACTION

Anonymous authors
Paper under double-blind review

ABSTRACT

A network can memorise training examples one by one, or learn a rule: a *pattern* shared by several examples, mapped to one label and applied to unseen inputs. How many examples a rule needs has two answers that seem to conflict. A backdoor, a trigger pattern added to training images relabelled to a target class, needs a roughly fixed *count* of these poisoned examples, whereas a class changes its prediction only when about half of it is relabelled, a *fraction*. We interpret both thresholds as a sum of two costs, counted in examples, and turn this into a cost model. A build cost separates the pattern from competing features, and counter-examples, images with the same pattern under their true label, add a weighted cost that separates it from competing labels. The model gives both regimes as limits: a count when nothing contradicts the pattern, and one half inside a class, whose correctly labelled images are all counter-examples. It also makes four predictions: counter-examples leave the build cost unchanged, the threshold is linear in them, different kinds add, and a backdoor forms at parity, against an equal number of counter-examples, only if each cancels less than one poisoned example. We test these predictions with a small synthetic patch as the pattern. All four hold, and both sides of parity occur, depending on how predictive another feature in the image is, while the pattern moves mainly the build cost. We therefore read the constants as properties of the network's representation, and expect the equation's form to transfer.

1 INTRODUCTION

A network can lower its training loss by memorising an example or by learning a rule that covers it: a pattern shared by several examples, mapped to their label and applied to unseen inputs. Both happen, and memorising rare examples can be necessary on long-tailed data (Feldman, 2020; Feldman & Zhang, 2020). An example stops being rare once enough others share its pattern, but *how many* examples are enough remains poorly understood.

Two lines of work have addressed this question, but they disagree on the form the answer takes. In data poisoning, a backdoor is made by adding a trigger pattern, such as a small patch, to some training images and relabelling them to a target class. The number of these *poisoned examples* a backdoor needs is close to a *count*: it changes little as the corpus grows (Souly et al., 2025), and in image classifiers a handful is enough (Yang et al., 2022). In the other line of work, label noise, relabelling examples within a class does not change the class prediction until the relabelled examples make up roughly half the class, which is the condition under which the clean Bayes label survives class-conditional noise (Natarajan et al., 2013; Patrini et al., 2017). Both describe a *threshold*, the number of examples at which a network starts to apply a rule to unseen inputs, but a fixed count and a fixed fraction of a growing class scale differently with dataset size, and existing accounts do not explain how both can emerge within a single learning mechanism.

We propose an interpretation of both thresholds as a sum of two costs, counted in examples. Consider a training set with k poisoned examples and m *counter-examples*, images that carry the same pattern under their true label. The threshold k^* is the number of poisoned examples at which the network starts to label unseen images carrying the pattern with the target instead of their own class, and our reading gives one equation for both regimes:

$$k^* = k_0 + \bar{w}\, m.$$

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

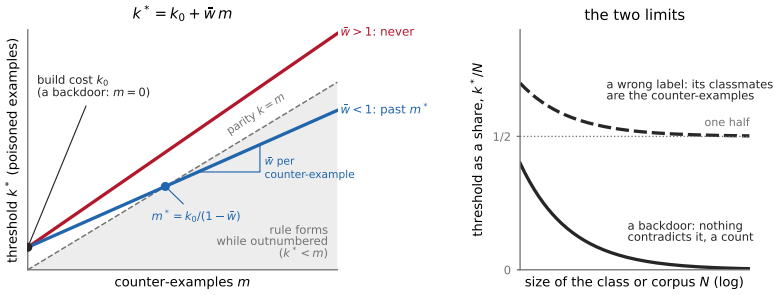

Figure 1: The threshold equation visualized (schematic). *Left*: the threshold k^* against the number of counter-examples m, with the build cost k_0 as intercept and the weight $\bar{w}$ as slope. Below the parity line $k = m$, the rule forms with fewer poisoned examples than counter-examples, which only a weight below one reaches, past $m^* = k_0/(1-\bar{w})$.. *Right*: the two limits as a share of the class or corpus. A backdoor's threshold is a count, so its share falls to zero, while a class's threshold tends to one half. The constants are for illustration purposes only.

The **build cost** k_0 is what it takes to separate the pattern from everything else the network represents. The **weight** $\bar{w}$ is how many poisoned examples one counter-example cancels. A backdoor trigger appears nowhere else, so $m = 0$ and the threshold is the count k_0. A wrong label is contradicted by its own classmates, so m grows with the class and the threshold tends to one half (Figure 1, right).

Concerning the weight, symmetric majority voting would give $\bar{w} = 1$, one wrong label against one right one. However, the measured weight is not always one, and that has a visible consequence (Figure 1, left). If $\bar{w} < 1$, a backdoor can form *while its poisoned examples are outnumbered* by counter-examples. If $\bar{w} > 1$, it does not form at parity, with as many counter-examples as poisoned examples, anywhere in the range we tested. The same trigger, a small patch of random colours, falls on both sides when we add it to two versions of CIFAR-10. In the *plain* set nothing else is changed, while in the *shortcut* set every image also carries two painted corners: corner A almost always shows a colour assigned to its class, an easy feature that predicts the label, and corner B a random colour. On the plain set the weight is 0.73 and the rule forms while outnumbered, whereas on the shortcut set it rises above one and the rule does not form at parity. When we keep the corners but vary how often corner A's colour matches the class, the weight rises with how predictive the corner is, crossing one (Figure 6, right), and the rule stops forming while outnumbered (Table 2; Section 6). The two terms also move separately, since changing only the pattern raises the build cost more than fivefold and leaves the weight almost unchanged.

Contributions.

- **One model for counts and fractions**: a threshold equation whose two limits are the backdoor count and the label-noise half (Section 2).
- **A measurement of both constants** from two sweeps, and a test of the four predictions that could falsify the model (Sections 4 and 5).
- **Counter-examples do not cancel one for one.** Whether correctly labelled copies of a pattern outvote a backdoor depends on the rest of the image, and on plain CIFAR-10 the backdoor forms while outnumbered (Sections 5 and 6).
- **The two constants move separately.** The build cost tracks how separable the pattern is from image content, and the weight how predictive another feature of the image is (Section 6).

2 THE MODEL

We place a pattern c on k training images and relabel them to a target y. The network is trained until it fits every training image,, so training accuracy cannot distinguish memorisation from a rule "$c \Rightarrow y$", but held-out images can. We therefore evaluate on *probes*, unseen images that carry c. The *target rate*, the attack success rate of the backdoor literature, is the share of probes predicted as y.

The *true-label accuracy* is the share predicted as their own class. As k grows the two trade places, and some seeds cross back and forth before they settle. We read k^* off a sweep over k as the count at which a logistic curve in $\log k$, fitted to their difference, crosses zero. The fit uses every count at which the network fits its training set, whereas the first count at which the target rate wins can be moved by a single noisy count or by a denser grid.

The Threshold model. We describe the transition with two effective quantities, both read against R_0, the representation learned without the poisoned examples. Building a detector for c and attaching it to y costs B, and each poisoned example that the rule then explains, instead of memorising it, brings a benefit u_+. A counter-example carries c under its true label, so the rule misclassifies it, and we let counter-example j cancel the benefit of $w_j \geq 0$ poisoned examples. The rule forms once $(k - \sum_j w_j)\, u_+ \geq B$, that is at

$$k^* \;=\; k_0 + \sum_{j=1}^{m} w_j, \qquad k_0 := \frac{B(c, y \mid R_0)}{u_+(c, y \mid R_0)}. \tag{1}$$

Operationally, the build cost k_0 is the threshold with no counter-examples, and the weight $\bar{w}$ is the slope of k^* on m for counter-examples drawn the same way, so that the sum equals $\bar{w}m$. We interpret k_0 as the cost of separating the pattern from the other *features* the network represents, and the sum as the cost of separating it from other *labels*.

The model has two limits. In a backdoor, $m = 0$ and $k^* = k_0$, a count that may grow with the corpus but far more slowly (Section 6). When k of the N images of a class are relabelled, the *mislabel* case, the pattern is the class's own appearance, which is not one feature but many, each with its own build cost $k_{0,i}$ and weight w_i. Every classmate that keeps its label is a counter-example, so $m = N - k$, and setting $k = k_i^*$ in Eq. (1) gives, for one feature,

$$k_i^* \;=\; \frac{k_{0,i} + w_i N}{1 + w_i}, \qquad \text{so} \qquad \frac{k_i^*}{N} \;\xrightarrow[N\to\infty]{}\; \frac{w_i}{1 + w_i}. \tag{2}$$

The build cost is divided by a growing class and drops out of the limit. Because the relabelled images are drawn at random from the class, a flipped image and a classmate differ only in their label, so by symmetry each cancels one of the other and $w_i \approx 1$, as the mislabel sweeps confirm(Table 2). Every feature then crosses at half the class, and so does any combination of the features' thresholds that lies between the smallest and the largest of them; how they combine at finite N is not modelled.

Eq. (1) makes four predictions, each a constraint that could fail:

(i) the build cost is the same with and without counter-examples;

(ii) one kind of counter-example has one weight, so k^* is linear in m;

(iii) the weights of different kinds add;

(iv) the weight decides whether poisoned examples form a rule at parity.

Only the ratio $k_0 = B/u_+$ is identified, and the model fixes the value of neither k_0 nor $\bar{w}$, which we measure.

3 Experimental setup

Three conditions. In the **backdoor** condition, a 6×6 patch of random colours, drawn once, is added to k images relabelled to one target. The **counter-example** condition adds m counter-examples from every class but the target, so they contradict the rule without founding a second one. In the **mislabel** condition no patch is added: some images of one class are relabelled, and the classmates that keep their label are the counter-examples. Appendix A shows an example of each.

Two training sets. The **plain** set is CIFAR-10 as it comes, and it carries every main result. The **shortcut** set is the same images with two painted corners, a standard way to add a cheap, predictive feature (Geirhos et al., 2020; Shah et al., 2020; Hermann et al., 2024; Suhail et al., 2025; Park et al., 2026). Corner A shows the image's class colour with probability 0.975, and corner B a random colour. We use it as a stress test.

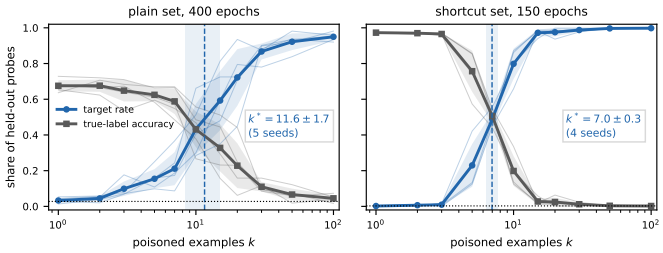

Figure 2: The threshold in the backdoor condition. Target rate and true-label accuracy on held-out probes, against the number of poisoned examples k. Thin lines are seeds, bands ± 1 s.d., and the dotted line is the target rate with no poisoned examples. The dashed line and shaded interval mark k^*, which here is the build cost k_0.

Training and estimation. The main model is a small CNN, trained on a class-balanced sample of CIFAR-10 until it fits every training image (Appendix A). Weights are least-squares slopes of k^* on m, fitted to the thresholds of individual seeds, and build costs are means over the seeds of the backdoor run; all intervals are 95% bootstrap intervals (Appendix B). A seed is included only if it passes a set of automated guards, for example that the network fits its whole training set (Appendix J). Appendix D traces every number to its run, and Appendix C lists every prediction we tested.

4 Testing the Model

We first check that k^* marks a real change in training and then test constraints (i)–(iii) in turn; constraint (iv) has its own section.

The threshold marks a change in training. If the network switches from memorising poisoned examples to a rule, it should fit them earlier and forget them less (Arpit et al., 2017; Toneva et al., 2019). This is what we observe: below the threshold, poisoned examples are fitted later and forgotten more than a size-matched control, and past it the order reverses (Appendix E).

(i) Counter-examples leave the build cost unchanged. Without counter-examples $k^* = k_0$ (Figure 2), and Eq. (1) says that extrapolating k^* back to $m = 0$ must return the same k_0. We fit a line to the sweep without its $m = 0$ cell and extrapolate it back to $m = 0$. On the plain set the build cost extrapolated from the sweep is 10.5 $[8.2, 12.9]$ examples, and the build cost measured without counter-examples is 11.6 $[8.9, 14.5]$. Their difference, -1.0 $[-4.7, +2.5]$, is consistent with zero, and five other plain-set measurements of the build cost, from 8.9 to 12.8, also fall inside the extrapolated interval. On the shortcut set the extrapolated build cost is 6.5 $[3.3, 10.0]$ against a measured 7.0 $[6.4, 7.6]$, a difference of -0.5 $[-3.9, +3.2]$. The test therefore resolves a change of the build cost of about four examples, a third of its plain value and half of its shortcut value, and a smaller change would pass unseen (Appendix B).

(ii) k^* is linear in m. Eq. (1) predicts $k^* - k_0 = \bar{w}m$, a line through the origin with no free intercept. We take k_0 from the backdoor run and, for each cell of the sweep, fit the weight on the other cells only, so that every cell is predicted by data it took no part in (Figure 3). All five plain-set cells fall inside the 95% interval of their prediction error, the largest miss being 1.3 standard errors. On the shortcut set four of five do, and the fifth, at $m = 100$, falls 7 examples below its prediction, 2.1 standard errors, and is drawn open in Figure 3. A weight fitted on the smallest sweeps alone ($m \leq 30$) also predicts every cell from $m = 60$ to 150, and a quadratic term is indistinguishable from zero over this fifteenfold range of m. The weight is $\bar{w} = 0.73$ on the plain set, with a 95% interval of 0.67 to 0.78, and 1.14 on the shortcut set, with 1.08 to 1.18, so both intervals exclude one.

(iii) Two kinds of counter-example add. If the sum in Eq. (1) runs over examples, a mix of two kinds must be priced by adding their contributions:

$$k^*_{\text{mix}} \;=\; k_0 \;+\; \bar{w}_A m_A \;+\; \bar{w}_B m_B. \tag{3}$$

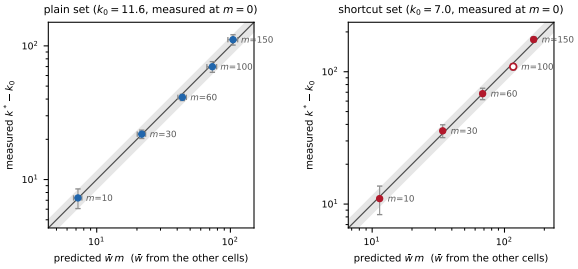

Figure 3: Counter-examples add linearly. Measured $k^* - k_0$ against the prediction $\bar{w}m$, one point per cell, with k_0 from a separate backdoor run and each cell's weight fitted on the other cells only. Bars are separate 95% intervals of the prediction (horizontal) and of the measured mean (vertical); the test uses the interval of their difference, which excludes zero only for the open marker. The line $y = x$ is the prediction of Eq. (1), and the $\pm 15\%$ band is for scale.

Table 1: Two kinds of counter-example add. Mixed cells, with m_A identical and m_B noisy counter-examples at noise ν, against the prediction of Eq. (3) and its two failure bounds, $w_B = 0$ and $w_B = w_A$. Mean $\pm$ s.e., seeds in brackets; dev. is the deviation from the prediction in standard errors.

						bounds		
set	m_A+m_B	ν	w_B	predicted	**measured**	w_B=0	w_B=w_A	dev.
PLAIN	30+30	0.2	0.616	52.0	$\mathbf{50.2 \pm 1.5}$ (9)	33.5	55.5	−1.2
	30+30	0.4	0.514	48.9	$\mathbf{49.2 \pm 2.3}$ (9)	33.5	55.5	+0.1
	60+30	0.2	0.616	73.9	$\mathbf{76.8 \pm 2.1}$ (11)	55.5	77.4	+1.3
	60+30	0.4	0.514	70.9	$\mathbf{74.6 \pm 1.8}$ (11)	55.5	77.4	+2.0
SHORTCUT	30+30	0.2	0.677	61.3	$\mathbf{65.0 \pm 2.5}$ (9)	41.0	75.0	+1.5
	30+30	0.4	0.473	55.2	$\mathbf{59.1 \pm 1.8}$ (10)	41.0	75.0	+2.3
	60+30	0.2	0.677	95.3	$\mathbf{95.0 \pm 3.1}$ (9)	75.0	109.0	−0.1
	60+30	0.4	0.473	89.2	$\mathbf{90.4 \pm 2.3}$ (12)	75.0	109.0	+0.5

Kind A carries the identical patch and kind B a noisy copy, with noise level ν. We measure $\bar{w}_A$ as the slope of the identical-patch sweep, 0.731 on the plain set and 1.133 on the shortcut set, where it also pools the identical-patch cells of the run that holds the mixed cells. We measure $\bar{w}_B$ in separate cells whose counter-examples are all noisy, and predict the mixed cells from these two weights and k_0 (Table 1). Every cell trains its own networks, so the mixed cells enter none of the inputs (Appendix D). The sum could fail in two directions, if noisy counter-examples cost nothing or as much as identical ones, and on our cells these two failures lie 22 to 34 examples apart. All eight mixed cells fall between them, each within 2.3 standard errors of the sum and six within 1.5. The deviations lean positive, by 0.8 standard errors on average, and the result holds with every guard switched off.

5 The weight is not always one

Eq. (1) leaves the weight free, and symmetric majority voting would set it to one. The measured weight is not one, and which side of one it falls on has a sharp consequence.

The weight decides whether the rule forms at parity. At parity, with as many poisoned examples as counter-examples, the rule forms if $m \geq k_0 + \bar{w}m$, that is, if $(1 - \bar{w})\, m \geq k_0$. Equivalently, the surplus of poisoned examples over counter-examples that the rule needs is

$$k^* - m \;=\; k_0 - (1 - \bar{w})\, m. \tag{4}$$

If $\bar{w} \geq 1$, each counter-example cancels at least one poisoned example and the build cost comes on top, so the surplus stays positive: the rule needs more poisoned examples than counter-examples, however many there are. If $\bar{w} < 1$, each counter-example cancels less than one poisoned example, and the shortfall, $1 - \bar{w}$ per counter-example, accumulates until it pays the build cost at $m^* = k_0/(1 - \bar{w})$. The rule forms while its poisoned examples are outnumbered.

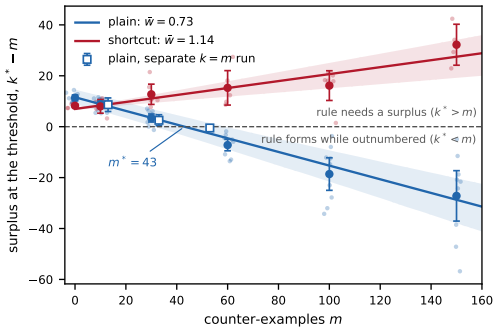

Figure 4: Weight influence in rule formation. The surplus $k^* - m$ at the threshold, against the number of counter-examples. Faint points are seeds and markers are cell means (95% interval). The lines are Eq. (4), with k_0 from the backdoor run and $\bar{w}$ from the sweep, and the bands bootstrap both. Open squares mark a separate plain-set run at $k = m$. Below the dashed line the rule forms with fewer poisoned examples than counter-examples.

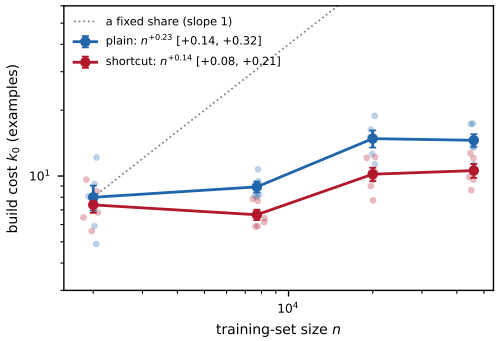

Figure 5: The build cost grows far more slowly than the corpus. Build cost against training-set size, backdoor condition, log–log. Faint points are seeds, markers are means, and the dotted line is a fixed share (slope one).

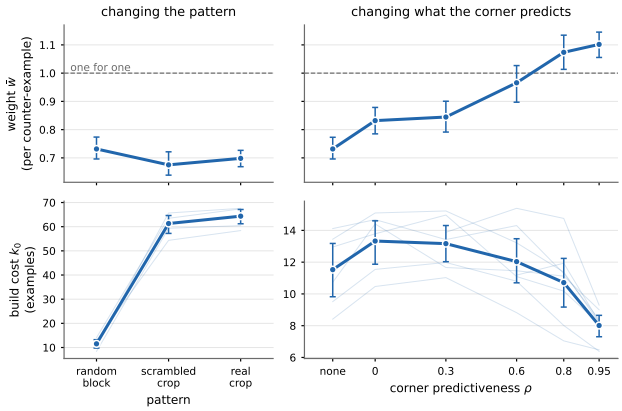

Figure 6: The two terms, moved one knob at a time. Weight (top, one scale for both panels) and build cost (bottom) on the plain set. Eight seeds per cell; intervals bootstrap seeds, and faint lines follow one seed. *Left*: only the pattern changes, with one fixed crop for every seed; drawing a new crop per seed gives lower counts (Table 2). *Right*: the same patch on the shortcut set's two corners, with corner A matching the class with probability $(1 + \rho)/2$ and corner B random; "none" has no corners.

The two sets fall on opposite sides (Figure 4). On the plain set, where $\bar{w} = 0.73$, the surplus changes sign at $m^* = 43$. From $m = 60$ upward, every plain seed forms the rule with fewer poisoned examples than counter-examples. On the shortcut set, where $\bar{w} = 1.14$, the surplus grows with m and stays positive in every seed. A separate run that sets $k = m$ directly agrees: at $k = m = 50$ the rule forms in five of eight plain seeds and in one of eight shortcut seeds (Appendix E).

6 What the build cost and the weight track

Our reading of Eq. (1) gives each term its own job: the build cost separates the pattern from the other features the network represents, and the weight separates it from other labels. If that reading holds, the two should respond to different changes of the data, and Table 2 lists which term each change moves.

The two terms can move independently. Changing only the pattern, from a random block to a real image crop or the same crop scrambled, raises the build cost more than fivefold while the weight

stays almost flat, within its uncertainty (Figure 6, left). Changing only the counter-examples leaves the build cost unchanged (Section 4), and degrading their copy of the pattern moves the weight. The shortcut set's corner, swept over its predictiveness ρ, moves both (Figure 6, right), so the two terms can be moved separately but are not independent in general.

The build cost is not set by salience. Across patterns the build cost spans an order of magnitude, and visibility does not order it: on the shortcut set, a whole-image watermark that a person can barely see is cheaper than the visible corner patch. Among three 6×6 patterns that change the image by the same amount, a random block, a real crop and the same crop scrambled, the real crop needs nearly four times as many examples as the random block. The random block is easy to separate even in the raw pixels, where its Mahalanobis distance from its window is 45 times that of a typical window, whereas the crops are not, at 4 and 1.1 times; this input-space statistic orders the three patterns but does not predict their cost. We read the build cost as the cost of separating the pattern in the network's representation, which an input statistic only partly reflects, and removing the background image, which lowers the cost about sevenfold, is consistent with that reading (Table 2).

The build cost grows far more slowly than the corpus. Growing the plain training set 23-fold only doubles the build cost (Figure 5), whereas a fixed share would grow with the corpus. A power law fitted over this range gives $n^{0.23}$, but four corpus sizes cannot tell slow growth from a step between flat stretches. On ImageNet32 with a ResNet-18 the shortcut-set build cost is still rising at half a million images, where it is about 25 examples (Table 2). A nearest-neighbour baseline, which memorises every example and learns no features, instead needs about 290 poisoned examples, grows almost in proportion to the corpus, and barely responds to counter-examples (Appendix H), so the count regime is consistent with the network learning a feature for the pattern.

The weight tracks another predictive feature. The plain and shortcut sets carry the same patch, yet their weights fall on opposite sides of one, and we hypothesised that a corner predictive of the class makes each counter-example cancel more, since it agrees with the counter-example's label and contradicts every poisoned image. To test this, we keep both painted corners and vary how often corner A matches the class, with probability $(1+\rho)/2$, from 0.5 at $\rho=0$ to the shortcut set's 0.975 at $\rho=0.95$ (Figure 6, right). The weight rises monotonically with ρ, from 0.73 without a corner to 1.10 at the most predictive corner, crossing one between $\rho=0.6$ and 0.8, and the rule stops forming while outnumbered (Table 2). The experiment does not separate this reading from a smaller benefit per poisoned example, since poisoned images also carry a corner for their true class.

The weight also tracks how closely a counter-example matches. Adding independent noise to the patch on the counter-examples only, which keeps its mean and colours and degrades only its match, lowers the weight gradually, to about half on the shortcut set (Table 2). The network still predicts the target for a quarter to half of the probes carrying the noisy patch, so a counter-example can remain recognisably the same pattern and still cancel only half a poisoned example. The weight thus follows how well the counter-examples match the planted pattern, not how separable that pattern is from image content, which moves the build cost instead.

7 A CLASS: ONE HALF

In the mislabel condition the threshold should approach half the class (Section 2), and it does. With the class pair fixed, the threshold is a straight line in the class size N, with slope 0.51 on both sets (Table 2). We also held the counter-examples fixed, by adding relabelled images to a class of N correctly labelled ones. The class then flips once the relabelled images match the correct ones one for one (Table 2).

One half is what class-conditional noise already implies (Natarajan et al., 2013; Patrini et al., 2017), and the equation recovers it as a limit. A class is many features, each build cost is divided away as the class grows, and classmates cancel each feature one for one. The model claims only this limit: the intercepts of both sweeps are finite-size terms of a feature combination it does not describe, and we do not read them as a build cost.

Table 2: What each change moves. Plain set unless marked (s) for the shortcut set, SmallCNN on CIFAR-10 with $n = 7{,}700$; build costs are read in the backdoor condition. Arrows follow the change in the first column, and "not measured" marks runs without counter-examples. At the two fixed crops, the build cost extrapolated from $m \geq 30$ is 60.8 and 64.4, against 61.3 and 64.4 measured. [a] Medians, with each seed drawing its own crop; separability is the pattern's Mahalanobis distance from its window, relative to a typical window. [b] Read at half the target rate's ceiling, the only readout without a true class. Runs and seeds are in Appendix D.

(a) Changes, and the term each one moves

change	build cost k_0	weight $\bar{w}$	moves
pattern: random block $\rightarrow$ scrambled crop $\rightarrow$ real crop, one fixed crop (Fig. 6)	$11.5 \rightarrow 61 \rightarrow 64$	$0.73 \rightarrow 0.68 \rightarrow 0.70$	**build cost**
the same patterns, one crop per seed; separability $45\times \rightarrow 4\times \rightarrow 1.1\times$ [a]	$11 \rightarrow 29 \rightarrow 38$	not measured	**build cost**
other patterns: 3×3 block; faint block; (s) watermark vs. block	23; ≈ 80; (s) 4–7 vs. 8.5	not measured	**build cost**
image replaced by a blank field [b]	$15 \rightarrow 2$	not measured	**build cost**
corpus: n from 2k to 46k, CIFAR-10 (Fig. 5)	$\propto n^{0.23}$; (s) $n^{0.14}$	not measured	**build cost**, sublinearly
corpus: (s) n from 46k to 500k, ImageNet32, ResNet-18	$15 \rightarrow 17 \rightarrow \approx 25$	not measured	**build cost**, sublinearly
number of counter-examples, m from 0 to 150 (Section 4)	unchanged	one value, linear in m	**neither**
noise on the counter-examples' patch, ν from 0 to 0.4	unchanged (shared $m = 0$ run)	$\times 0.70$; (s) $\times 0.47$	**weight**
painted corner, predictiveness ρ: none $\rightarrow 0 \rightarrow 0.95$ (Fig. 6)	$11.5 \rightarrow 13.3 \rightarrow 8.0$	$0.73 \rightarrow 0.83 \rightarrow 1.10$	**both**
same sweep: seeds forming the rule while outnumbered at $m = 100$	$5/5 \rightarrow 5/7 \rightarrow 0/7$, and already $0/7$ at $\rho = 0.8$		**side of parity**
bottleneck width 8 vs. 32	$\times 0.99$; (s) $\times 0.98$	$\times 1.00$; (s) $\times 1.04$	**neither**

(b) Limits and baseline

setting	quantity	value
mislabel, class size N up to 780	slope of k^* on N	0.512; (s) 0.509
mislabel, counter-examples held fixed, two class pairs	slope of k^* on N	0.956; 0.975
planted patch's constants carried to a class, $N = 300$	margin, predicted vs. measured	-33 vs. $+4$, $+17$; (s) $+26$ vs. $+14$, $+5$
1-nearest neighbour in pixel space, no learned features	k_0; growth in n; slope in m	286; $n^{0.74}$; 0.31

8 LIMITATIONS

What transfers. Within one pattern on one set of images, the constants carry across the sweep (Section 4). Capacity moves neither term: over a fourfold range of bottleneck widths both stay within 15% (Table 2). Changing the pattern, the set, the corpus or the architecture moves the build cost, and changing the representation moves the weight too. A planted patch's constants do not predict a class (Table 2), so what transfers is the equation, and its constants must be measured where they are used.

Budgets and architectures. Each set is read once it fits its training set, but the architecture comparisons ran at other budgets, so across families we claim orderings, not levels. Beyond CIFAR-10 only the backdoor condition is tested.

How k^* is read. k^* spreads more across seeds as m grows, from a standard deviation of about 2 examples at $m = 0$ to 16 at $m = 150$ on the plain set, and seeds that cross back and forth give different values at their first and last crossings. Reading those edges instead of the fitted zero keeps

each weight on its side of one and passes the build-cost and additivity checks, but moves the plain crossover m^* from 43 to between 38 and 60, so per-seed counts near it depend on the readout (Appendix J).

Scope. Eq. (1) is an empirical accounting, and we make no claim about language-model pretraining or about re-pointing a detector that already exists.

9 Related work

Memorisation and frequency thresholds. Feldman (2020); Feldman & Zhang (2020) show why rare examples may need to be memorised, whereas we plant a subpopulation at a controlled count and read when it transfers. Jiang et al. (2021) vary training-set size and measure consistency. Memorisation varies across model sizes (Lukasik et al., 2024), whereas our build cost stays flat over bottleneck widths. Predictivity–availability trade-offs (Hermann et al., 2024; Qiu et al., 2024; Sagawa et al., 2020) and learning-order results (Kalimeris et al., 2019; Arpit et al., 2017; Zhang et al., 2017) give context for our ordering of costs. Frequency also sets quantization (Michaud et al., 2023), knowledge capacity (Allen-Zhu & Li, 2024; Zucchet et al., 2025), duplication effects (Kandpal et al., 2022; Lee et al., 2022; Carlini et al., 2023) and concept frequency (Udandarao et al., 2024). Farné et al. (2026) study structured rules with randomly labelled exceptions, while we measure how labelled counter-examples shift a pattern's threshold. The critical data size of Zhu et al. (2024) concerns a whole-task transition under joint scaling. The balance between memorisation and generalisation also shifts with data size (So et al., 2026; Singh et al., 2025), and the two can coexist under heavy label noise (Liu & Lu, 2026).

Poisoning: a count or a fraction. Souly et al. (2025) establish the constant count in language models. In vision, Schwarzschild et al. (2021) hold the poisoned share fixed while varying training-set size and find no consistent trend. They conclude that a percentage budget does not compare across corpus sizes, which is the shape of Eq. (1) with no counter-examples. Carlini et al. (2024) show that poisoning a small share of a web-scale corpus is practical and cheap. Dirty-label backdoors (Gu et al., 2017) and clean-label ones (Turner et al., 2019) differ in whether poisoned examples are relabelled, which Eq. (1) reads as a change in the counter-example term. Our mislabel condition is class-conditional pair-flip noise, whose clean label survives below a flip rate of one half (Natarajan et al., 2013; Patrini et al., 2017). We claim no new threshold there, only that Eq. (2) recovers that half as the limit of a class of features.

What the representation already holds. Backgrounds alone carry accuracy (Xiao et al., 2021), models use spurious features at scale (Singla & Feizi, 2022), networks prefer the simplest sufficient feature (Shah et al., 2020; Geirhos et al., 2019), and a cue's precision shapes how much a classifier relies on it (Park et al., 2026). Burgert et al. (2025) suppress one feature at a time, the same manipulation as replacing the image with a blank field (Table 2). Closest to our build cost, Yang et al. (2022) report that a handful of planted rare patterns suffices. They do not locate the threshold or separate the effect of contradicting examples, which Eq. (1) addresses.

Conclusion

Backdoor poisoning and label noise are usually studied apart, and their thresholds are of different kinds: a count of poisoned examples for a backdoor, a fraction of the class for a wrong label. We propose to read both as a sum of two costs, counted in examples: a build cost that separates the pattern from the other features the network represents, and a weighted cost of counter-examples that separates it from competing labels. This reading gives a threshold model with both regimes as limits, a count when nothing contradicts the pattern and one half inside a class. On images with a planted pattern, its less obvious predictions hold: counter-examples leave the build cost unchanged, the two costs add, and the two terms can be moved separately, the pattern moving the build cost and the counter-examples' match moving the weight. The weight of a counter-example does not have to be one: it rises with how predictive another feature of the image is, and it decides whether poisoned examples can form a rule while outnumbered. We read the constants as properties of the network's representation and of the pattern read against it, so anything that changes either, the pattern, the data or the model, changes them. Our experiments are small and on images, and we offer the equation as a simplification to test at scale, not as an account of how networks learn rules.

AI USE STATEMENT

In this work we used generative AI tools as a research assistant with which ideas could be discussed and tested. We used them for the following tasks, which require disclosure: expressing the conceptual framework in mathematical form as an inequality; refining the hypothesis; proposing some experimental designs; several of which the authors later discarded or redesigned; implementing the guards, which the authors requested, and their failure conditions; and implementing the experiments, processing the results and assisting with their interpretation, including several audits, requested by the authors, that retracted or weakened earlier claims. AI tools were also used to help draft the text and for literature research. We have reviewed all AI-assisted work, and we established automated checks and guards to verify the data generated and reported. All experiments were run by the authors on an HPC cluster. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All training sets, guards and readouts are scripted. Every figure and table is built by a script from the stored per-seed results, and every stored result is re-checked by one automated guard checker (Appendix J); Appendix D names the run behind each headline number. The code, the stored results and the build scripts will be released with the paper.

ETHICS STATEMENT

This work studies how many poisoned examples a backdoor needs, which could inform attackers as well as defenders. All experiments use public image data sets at small scale, no model or data set is released with a backdoor, and the account is meant to help defenders reason about which training data can install a rule.

REFERENCES

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.3, knowledge capacity scaling laws. *arXiv preprint arXiv:2404.05405*, 2024.

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In *International Conference on Machine Learning (ICML)*, 2017. arXiv:1706.05394.

Tom Burgert, Oliver Stoll, Paolo Rota, and Begüm Demir. ImageNet-trained CNNs are not biased towards texture: Revisiting feature reliance through controlled suppression. *arXiv preprint arXiv:2509.20234*, 2025.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. Quantifying memorization across neural language models. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2202.07646.

Nicholas Carlini, Matthew Jagielski, Christopher A. Choquette-Choo, Daniel Paleka, Will Pearce, Hyrum S. Anderson, Andreas Terzis, Kurt Thomas, and Florian Tramèr. Poisoning web-scale training datasets is practical. In *IEEE Symposium on Security and Privacy (S&P)*, pp. 407–425, 2024. arXiv:2302.10149.

Gabriele Farné, Fabrizio Boncoraglio, and Lenka Zdeborová. The rules-and-facts model for simultaneous generalization and memorization in neural networks. *arXiv preprint arXiv:2603.25579*, 2026.

Vitaly Feldman. Does learning require memorization? A short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2020. arXiv:1906.05271.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. arXiv:2008.03703.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations (ICLR)*, 2019. arXiv:1811.12231.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. In *Nature Machine Intelligence*, 2020. arXiv:2004.07780.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Katherine Hermann, Hossein Mobahi, Thomas Fel, and Michael C. Mozer. On the foundations of shortcut learning. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.16228.

Ziheng Jiang, Chiyuan Zhang, Kunal Talwar, and Michael C. Mozer. Characterizing structural regularities of labeled data in overparameterized models. In *International Conference on Machine Learning (ICML)*, 2021. arXiv:2002.03206.

Dimitris Kalimeris, Gal Kaplun, Preetum Nakkiran, Benjamin L. Edelman, Tristan Yang, Boaz Barak, and Haofeng Zhang. SGD on neural networks learns functions of increasing complexity. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. arXiv:1905.11604.

Nikhil Kandpal, Eric Wallace, and Colin Raffel. Deduplicating training data mitigates privacy risks in language models. In *International Conference on Machine Learning (ICML)*, 2022. arXiv:2202.06539.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022. arXiv:2107.06499.

Linyu Liu and Pinyan Lu. Unveiling memorization–generalization coexistence: A case study on arithmetic tasks with label noise. *arXiv preprint arXiv:2605.18022*, 2026.

Michal Lukasik, Vaishnavh Nagarajan, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. What do larger image classifiers memorise? *Transactions on Machine Learning Research*, 2024. arXiv preprint arXiv:2310.05337.

Eric J. Michaud, Ziming Liu, Uzay Girit, and Max Tegmark. The quantization model of neural scaling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2303.13506.

Nagarajan Natarajan, Inderjit S. Dhillon, Pradeep Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

Chanho Park, Woochan Lee, Janyeong Oh, Geongho Gong, Minshu Kim, Yeachan Kwak, and Seongim Choi. Early cue precision shapes visual shortcut learning in controlled cue-manipulation benchmarks. *arXiv preprint arXiv:2606.30344*, 2026.

Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. arXiv:1609.03683.

GuanWen Qiu, Da Kuang, and Surbhi Goel. Complexity matters: Dynamics of feature learning in the presence of spurious correlations. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2403.03375.

Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. An investigation of why overparameterization exacerbates spurious correlations. In *International Conference on Machine Learning (ICML)*, 2020. arXiv:2005.04345.

Avi Schwarzschild, Micah Goldblum, Arjun Gupta, John P. Dickerson, and Tom Goldstein. Just how toxic is data poisoning? a unified benchmark for backdoor and data poisoning attacks. In *International Conference on Machine Learning (ICML)*, 2021. arXiv:2006.12557.

Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. arXiv:2006.07710.

Vaibhav Singh, Eugene Belilovsky, and Rahaf Aljundi. When data falls short: Grokking below the critical threshold. *arXiv preprint arXiv:2511.04760*, 2025.

Sahil Singla and Soheil Feizi. Salient ImageNet: How to discover spurious features in deep learning? In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2110.04301.

Shin So, Kyelim Lee, and Albert No. Slower generalization, faster memorization: A sweet spot in algorithmic learning. *arXiv preprint arXiv:2605.14659*, 2026.

Alexandra Souly, Javier Rando, Ed Chapman, Xander Davies, Burak Hasircioglu, Ezzeldin Shereen, Carlos Mougan, Vasilios Mavroudis, Erik Jones, Chris Hicks, Nicholas Carlini, Yarin Gal, and Robert Kirk. Poisoning attacks on LLMs require a near-constant number of poison samples. *arXiv preprint arXiv:2510.07192*, 2025.

Pirzada Suhail, Vrinda Goel, and Amit Sethi. Shortcut learning susceptibility in vision classifiers. *arXiv preprint arXiv:2502.09150*, 2025.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations (ICLR)*, 2019. arXiv:1812.05159.

Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*, 2019.

Vishaal Udandarao, Ameya Prabhu, Adhiraj Ghosh, Yash Sharma, Philip H. S. Torr, Adel Bibi, Samuel Albanie, and Matthias Bethge. No "zero-shot" without exponential data: Pretraining concept frequency determines multimodal model performance. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2404.04125.

Kai Xiao, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. Noise or signal: The role of image backgrounds in object recognition. In *International Conference on Learning Representations (ICLR)*, 2021. arXiv:2006.09994.

Yao-Yuan Yang, Chi-Ning Chou, and Kamalika Chaudhuri. Understanding rare spurious correlations in neural networks. In *ICML 2022 Workshop on Spurious Correlations, Invariance and Stability*, 2022. arXiv:2202.05189.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, 2017. arXiv:1611.03530.

Xuekai Zhu, Yao Fu, Bowen Zhou, and Zhouhan Lin. Critical data size of language models from a grokking perspective. *arXiv preprint arXiv:2401.10463*, 2024.

Nicolas Zucchet, Jörg Bornschein, Stephanie Chan, Andrew Lampinen, Razvan Pascanu, and Soham De. How do language models learn facts? dynamics, curricula and hallucinations. *arXiv preprint arXiv:2503.21676*, 2025.

APPENDIX A EXPERIMENTAL DETAILS

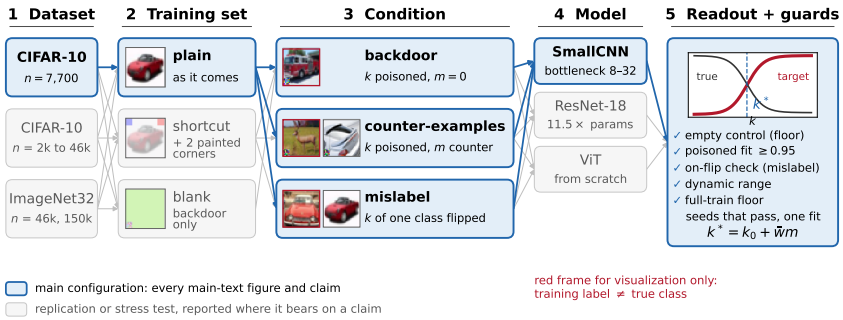

Figure 7: The experimental framework. Every run combines one choice per column and reads k^* (schematic). A run counts only if it passes every guard. Blue: the main configuration, behind every main-text figure and claim. Grey: replications and stress tests.

Data and models. The main configuration trains on a class-balanced sample of CIFAR-10: $n = 7{,}700$ images, about 770 per class. Each cell is read on 600 held-out probes. The network is a small CNN with four convolutional blocks of width 64, global average pooling, a bottleneck of width 32, and a linear classifier. The capacity sweep (Appendix E) varies the bottleneck width. Replications use ResNet-18 from scratch, with the standard 3×3 stride-1 stem for 32×32 inputs and no bottleneck. They also use a vision transformer from scratch (4×4 patches, 64 tokens, width 128, depth 4, 4 heads, mean pooling). The corpus-size sweep uses $n = 2{,}000$ to $46{,}000$ CIFAR-10 images on both sets. The scale replication uses ImageNet32 at $n = 46{,}000$, $150{,}000$ and $500{,}000$ (Appendix I). The CNNs are trained with Adam and the transformer with AdamW, at learning rate 10^{-3} and batch size 128, with no augmentation.

Budgets. Each set is trained until it fits its training set, not for a common number of epochs. The shortcut set needs 150 epochs and the plain set 400. The vision transformer needs 600 epochs on CIFAR-10. On ImageNet32, ResNet-18 is trained for 120 epochs and the transformer for 240.

The training sets, drawn. Figure 8 shows one example of each condition on each set, generated by the code that builds them. The plain and shortcut sets are the same draws and differ only by the painted corners. Corner A carries the colour of the image's true class with probability 0.975. Corner B carries a colour drawn independently of the label. The corner sweep of Section 6 paints the same two corners but varies corner A's agreement, $(1 + \rho)/2$ with ρ from 0 to 0.95, so that even its least predictive corner shows the true class half the time, against a chance level of one in ten. A third training set, the *blank* set, replaces every image with a uniform field of a colour drawn independently of the label, and places the patch on that field. With no host image there is nothing for the pattern to compete with and no true class to predict, so the blank set is used only in the backdoor condition, to measure the build cost with the host image removed (Table 9).

The pattern and the counter-examples. Unless stated otherwise the pattern is PATCH6: a fixed uniform-random RGB 6×6 block, opaque, in the bottom-left corner. It is drawn once and pinned across seeds, and so is the target class. Counter-examples are drawn from the same pool as the poisoned examples: every class but the target, disjoint from them. They carry the identical pattern and keep their true labels, so only the label tells the two apart. Spreading them over classes lets them contradict the rule without founding a second one. A single-class set would teach "pattern $\Rightarrow$ that class". Spread over classes, they teach only "pattern $\Rightarrow$ true class", which is the identity. Noisy counter-examples (Section 6) carry the pattern plus independent zero-mean noise, drawn per image, with variance ν times the pattern's own.

Readout. Each cell is read on held-out probes that carry the pattern. The *target rate*, the attack success rate of the backdoor literature, is the share predicted as the target. The *true-label accuracy* is the share predicted as the image's own class. Per seed, k^* is the zero of a logistic in $\log k$, $d(k) = L + (U - L)/(1 + e^{-(\log k - \mu)/s})$. It is fitted by least squares to the margin $d =$ target rate

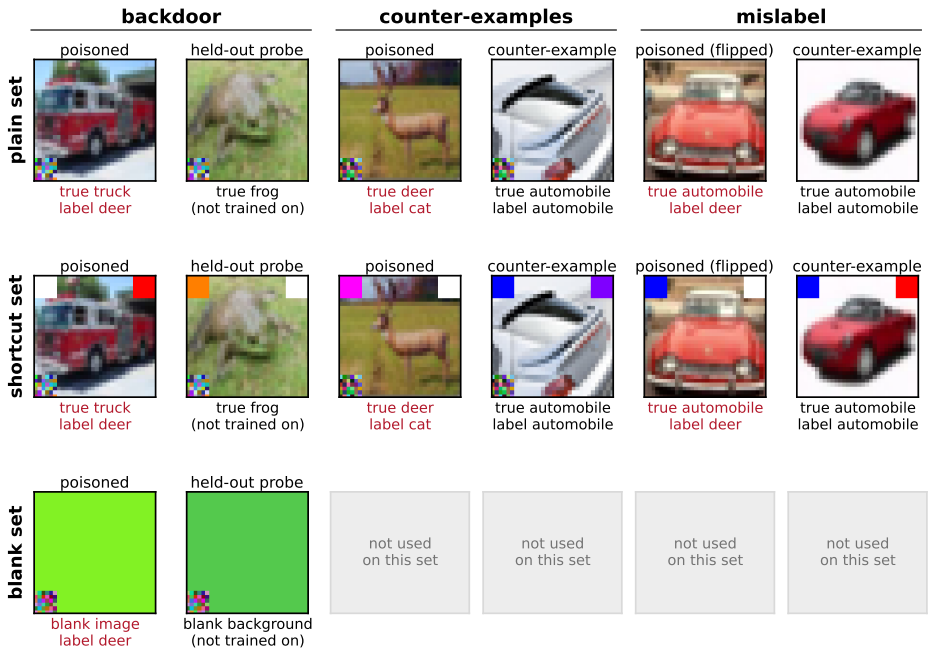

Figure 8: The three conditions on the three training sets. *Backdoor*: a poisoned example and a held-out probe. *Counter-examples*: a poisoned example and a counter-example, which carries the same patch under its own true label. *Mislabel*: a flipped image (the poisoned example) and a classmate that keeps its label (the counter-example); automobile is relabelled deer. Rows: the plain set; the shortcut set, the same images with two painted corners; the blank set. Red text: the training label differs from the true class. The pattern is the 6×6 patch at the bottom left.

$-$ true-label accuracy, over every count on the grid. Counts at which that seed's whole-set training accuracy is below 0.95 are left out of the fit (Appendix J). In the mislabel condition, a straight line fitted to the same margin over a band of counts around $N/2$ locates the zero. On the blank set there is no true class, so the threshold is read where the target rate crosses half its own ceiling. Every counter-example cell runs in two stages: a coarse grid spanning more than a factor of ten locates the crossing, and a refined grid is then placed around the crossings the seeds actually showed. The coarse grids were scaled to a prior estimate of k^*, but at $m = 150$ they cover every slope from 0 to about 1.5, so a threshold that did not grow with m, or grew faster than the line, would have been read as such.

Estimation. Every slope and every extrapolated build cost is a least-squares fit to the seeds that pass the guards, each seed in each cell an individual point, every measured build cost is a mean over seeds, and Appendix B gives how every interval in the paper is computed.

Appendix B Uncertainty

Every interval in the paper is a 95% interval, and every bootstrap draws $4{,}000$ resamples and reports the 2.5th and 97.5th percentiles, so intervals can be asymmetric. The unit of variation is the seed: each seed trains a new network and redraws which images are poisoned, so seed-to-seed spread includes both. Table 3 lists, for each quantity, how its point value and its interval are obtained.

What each test can resolve. Test (i) compares two uncertain numbers, so its resolution is the interval of their difference, about ±3.6 examples on both sets: a third of the plain build cost and half of the shortcut one. A change of the build cost smaller than that would pass unseen. Test (iii) is sharper: its two ways to fail lie 22 to 34 examples apart, while the standard errors of the mixed cells are 1.5 to 3.1. In test (ii) the held-out intervals widen with m, because the spread of k^* grows from about 2 examples at $m = 0$ to about 16 at $m = 150$ on the plain set.

Table 3: How each uncertainty is computed. A point is one seed in one cell of a sweep. Resampled units are drawn with replacement, and where two units are listed they are resampled independently.

quantity	point value	interval	resampled
k^* of one seed	zero of the logistic fitted to that seed's margin (Appendix A)	none; its spread across seeds enters every quantity below	—
build cost k_0	mean over the seeds of the backdoor run	$\pm$ s.e. in the text and Figure 2, whose shading is ± 1.96 s.e.; bootstrap: $[8.9, 14.5]$ plain, $[6.4, 7.6]$ shortcut	seeds
weight $\bar{w}$	least-squares slope of k^* on m over all points of the sweep	percentile bootstrap	points
extrapolated build cost, test (i)	the same fit without the $m = 0$ cell, read at $m = 0$	percentile bootstrap	points
extrapolated minus measured build cost, test (i)	difference of the two	percentile bootstrap: -1.0 $[-4.7, +2.5]$ plain, -0.5 $[-3.9, +3.2]$ shortcut	points; k_0 seeds
held-out cell, test (ii)	cell mean minus $k_0 + \bar{w}_{-m} m$, with $\bar{w}_{-m}$ fitted through the origin on the other cells	percentile bootstrap of that error; z is the error over its bootstrap standard deviation	k_0 seeds; other cells' points; the cell's seeds
mixed cell, test (iii)	mean over seeds, against $k_0 + \bar{w}_A m_A + \bar{w}_B m_B$; $\bar{w}_B$ is the mean of $(k^* - k_0)/m$ over the points of its pure cells	$\pm$ s.e. of the measured mean only, so the deviation in Table 1 ignores the prediction's own uncertainty and overstates any miss	—
crossover m^*, test (iv)	$k_0/(1 - \bar{w})$	percentile bootstrap: 43 $[31, 59]$	k_0 seeds; points
seeds on each side of parity	counts	none	—
weight and build cost in Figure 6	as above, per pattern and per ρ	percentile bootstrap	whole seeds, each with its full sweep
exponent of k_0 in n	least-squares slope of $\log k^*$ on $\log n$	percentile bootstrap	points
class slopes (Appendix G)	least-squares slope of k^* on N	percentile bootstrap	seeds within each N

What the intervals leave out. They do not include the choice of readout, which Table 12 varies, or the choice of guards, which Table 11 varies. They treat points as independent, although most seeds appear in several cells; resampling whole seeds instead gives the same intervals for the weights, $[0.67, 0.79]$ plain and $[1.09, 1.18]$ shortcut. The estimator does not impose the line that Eq. (1) predicts: each threshold is read from its own seed and cell, by a fit in k that knows nothing of m, and linearity in m is then tested on cells left out of the fit, by extrapolation from $m \leq 30$, and on mixed cells never fitted at all (Section 4). Fitting cell means instead gives weights of 0.732 (plain) and 1.145 (shortcut), and weighting them by inverse variance gives 0.708 and 1.123, so no conclusion depends on the estimator.

APPENDIX C EVERY PREDICTION TESTED

Every prediction of the equation that we tested, in the order the main text takes them. Some predictions for the mislabel condition assumed that a class has a single build cost and weight, before we recognised that a class is a bundle of features. They are reported, with their outcomes, in Appendix G, not here.

Table 4: Every prediction tested, and its outcome. Each prediction follows from Eq. (1). Plain set at 400 epochs, shortcut set at 150, SmallCNN, $n = 7{,}700$, unless stated. Intervals are 95%.

prediction	measured	outcome
(i) the build cost extrapolated from the counter-example sweep equals the measured one (§4)	extrapolated 10.5 [8.2, 12.9] vs. measured 11.6 (plain); 6.5 [3.3, 10.0] vs. 7.0 (shortcut); five other plain measurements, 8.9–12.8, all inside	pass
(ii) k^* linear in m, no free intercept (§4)	leave-one-out: $5/5$ cells in interval (plain, $\lvert z\rvert \le 1.3$), $4/5$ (shortcut; $m = 100$ at $z = -2.1$); quadratic term ≈ 0 over a fifteenfold span of m	pass
(ii) $\bar{w}$ fitted on $m \le 30$ predicts $m = 60$–150	$3/3$ cells in interval on both sets	pass
(iii) two kinds of counter-example add (§4)	$8/8$ mixed cells inside both bounds, within 2.3 s.e., six within 1.5; mean deviation $+0.8$ s.e.	pass
(iv) $\bar{w} < 1$: the rule forms while outnumbered past m^*; $\bar{w} > 1$: parity never suffices (§5)	plain: $m^* = 43$ [31, 59], and $31/31$ seeds at $m \ge 60$ form the rule with $k < m$; shortcut: surplus positive in $26/26$	pass
(iv) the same, in a separate $k = m$ run	plain $0/8$, $3/8$, $5/8$ at $m = 10, 30, 50$ (predicted surplus $+8.9, +3.5, -1.9$); shortcut $0/8$, $0/8$, $1/8$	pass
$\bar{w}$ rises with corner predictiveness ρ and crosses one (§6)	$0.83, 0.85, 0.97, 1.07, 1.10$ for $\rho = 0$–0.95, from 0.73 with no corner; slope $+0.32$ $[+0.24, +0.38]$	pass
capacity moves neither term by more than 15% (§8)	widths 8–32: ratios 0.98–1.06; intervals wide	pass
constants belong to a pattern: the planted patch's k_0, $\bar{w}$ do not predict the mislabel margin (§8)	misses by 10–50 examples, outside all four intervals	pass
a class tends to half its size: k^* proportional to N, slope near $1/2$ (§7)	slope 0.512 (plain, $N = 150$–780) and 0.509 (shortcut); with the counter-examples held fixed, $k^* \approx N$	pass

APPENDIX D PROVENANCE

Every headline number, the run it is read from, and that run's budget and seeds. PATCH6 is the pattern throughout unless the row says otherwise. Seeds: run (passing the guards).

quantity (where it is used)	condition	budget	seeds (passing)
k_0, 11.6 plain / 7.0 shortcut (§4, Fig. 2)	backdoor	400 / 150 ep.	5 (5) / 5 (4)
the same, plain, in the other runs: 12.8 (Table 5), 11.3 and 11.5 (the $m = 0$ cells of the counter-example and ρ sweeps), 8.9 (the $n = 7{,}700$ point of Fig. 5)	backdoor	400 ep.	8 (6); 16 (10); 8 (6); 6 (5)
$\bar{w}$ and the counter-example sweep, 0.73 plain (§4, Fig. 3, 4)	counter-examples, $m \leq 150$	400 ep.	16 (9–14 per cell)
$\bar{w}$ and the counter-example sweep, 1.14 shortcut (same)	counter-examples, $m \leq 150$	150 ep.	6 (4–6 per cell)
the $k = m$ run (§5, Appendix E)	$k = m$	400 / 150 ep.	8 per cell
Table 1, plain: $\bar{w}_A$, $\bar{w}_B$, mixed cells; the plain noise gradient (§6)	counter-examples, identical and noisy	400 ep.	16 (9–11 per cell)
Table 1, shortcut; the shortcut noise gradient (Table 6)	counter-examples, identical and noisy	150 ep.	6–14 per cell
Training dynamics (Table 5)	backdoor	400 ep.	8 (6)
Patterns and separability (§6, Tables 7 and 8)	backdoor; PATCH3, FAINT6, PATCH1, SHUF6, CROP6	400 ep. plain, 150 shortcut	5–8
k_0 against corpus size (Fig. 5)	backdoor	400 ep. plain, 250 shortcut	6 per n (5–6)
Host removed (Table 9)	backdoor, blank set	400 ep.	8 (4–7)
Capacity (§8, Appendix E)	counter-examples, $m \in \{0, 30\}$	400 / 150 ep.	6
Mislabel fraction and margins (§7, §8, Appendix G)	mislabel	400 ep.	8
Mislabel with the counter-examples held fixed, $N \in \{200, 400, 770\}$ (§7, Appendix G)	mislabel, class-balanced, flipped images added	400 ep.	8 per N (6–8), two class pairs
$\bar{w}$ and k_0 against corner predictiveness (§6, Fig. 6, Table 10)	counter-examples, $m \leq 100$	400 ep.	8, paired across ρ
$\bar{w}$ and k_0 against the pattern: scrambled and real crop, one fixed crop (§6, Fig. 6, Table 10)	counter-examples, $m \leq 100$	400 ep.	8 (4–7 per cell)
ImageNet32 shortcut set to $n = 500{,}000$ (§6, Appendix I)	backdoor	120 ep.	3 per n
ImageNet32 (Appendix I)	backdoor	120 / 240 ep.	3
k-NN baseline (Appendix H)	backdoor and counter-examples	no training	3

APPENDIX E SUPPORTING MEASUREMENTS

Training dynamics. Table 5 follows the backdoor run on the plain set through training. Below the threshold, poisoned examples are fitted later than a size-matched control. The fitting order reverses between $k = 7$ and 10. Poisoned examples are also forgotten more than the control below the threshold. That signal crosses chance between $k = 10$ and 15, around this run's $k^* = 12.8 \pm 0.9$. The threshold is the same at 250 epochs (13.0).

Parity at $k = m$. A separate run places poisoned examples and counter-examples in exact balance, eight seeds per cell, each set at its budget. On the plain set the grid runs from $k = m - 12$ to $m + 12$ in steps of two, with a tail above. So the fitted margin also gives the surplus: $+8.6 \pm 1.4$, $+2.5 \pm 1.1$ and -0.5 ± 0.6 at $m = 10$, 30 and 50. The predicted surplus is $+8.9$, $+3.5$ and -1.9.

Table 5: Training dynamics across the threshold. Backdoor condition, plain set, SmallCNN, $n = 7{,}700$, 400 epochs; $k^* = 12.8 \pm 0.9$ on the six of eight seeds that pass the guards. *Control*: size-matched training images with no pattern. *Forgetting AUROC*: above 0.5 the poisoned examples are forgotten more than the control, below it less.

k	3	7	10	15	20	30	100	300
target rate	0.10	0.23	0.38	0.52	0.67	0.81	0.96	0.99
poisoned, fitted at epoch	308	344	311	287	283	256	149	101
control, fitted at epoch	281	306	331	301	307	293	296	295
forgetting AUROC	0.66	0.58	0.54	0.34	0.34	0.22	0.09	0.05

	plain set ($\bar{w} = 0.73$)			shortcut set ($\bar{w} = 1.14$)		
$k = m$	10	30	50	10	30	50
poisoned examples fitted	0.950	0.996	0.995	1.000	1.000	1.000
target rate	0.262	0.419	**0.455**	0.278	0.361	0.417
true-label accuracy	0.565	0.449	**0.438**	0.714	0.632	0.577
seeds where the rule forms	0/8	3/8	5/8	0/8	0/8	1/8

Noisy counter-examples. Table 6 gives the shortcut-set gradient. The plain set falls in the same order: to 0.84 of its identical-patch weight at $\nu = 0.2$, and to 0.70 at $\nu = 0.4$.

Table 6: The weight falls as the counter-examples' patch departs from the planted one. Shortcut set. ν is the noise added to the patch on the counter-examples only. Top row: the weight at $m = 30$ relative to the identical patch, from one copy of each seed. Bottom row: how often the network still answers with the target on a probe carrying the *noisy* patch, as the range over the seeds that pass at $m = 30$ and the largest count.

noise ν	0	0.01	0.02	0.05	0.10	0.15	0.20	0.40
$\bar{w}$, relative to $\nu = 0$ ($m = 30$)	1.000	0.984	0.944	0.941	0.793	0.755	0.634	0.469
rule fires on the noisy probe	—	0.82–0.93	0.82–0.94	0.80–0.89	0.76–0.89	0.70–0.87	0.41–0.70	0.26–0.48

Capacity. We run the counter-example condition at $m \in \{0, 30\}$ and $\nu \in \{0, 0.10\}$, with bottleneck widths 32, 16 and 8, six seeds per cell. The build cost is the $m = 0$ cell, and the weight is $(k^*(30) - k_0)/30$. The criterion is that the width-8 value lies within 15% of the width-32 value.

	plain set, 400 epochs			shortcut set, 150 epochs		
width	32	16	8	32	16	8
k_0	11.6	11.6	11.5	7.9	7.2	7.7
$\bar{w}$, $\nu = 0$	0.80	0.72	0.80	1.10	1.17	1.15
$\bar{w}$, $\nu = 0.10$	0.70	0.71	0.70	0.96	0.97	1.01
width 8 over 32: k_0; $\bar{w}$ at $\nu = 0, 0.10$	0.99; 1.00, 1.00			0.98; 1.04, 1.06		

The point estimates all lie within 15%, but the intervals on the ratios are wide: on the plain set 0.71–1.33 for k_0, and 0.79–1.26 for the weight at $\nu = 0.10$. They are wide because the width-8, $\nu = 0.10$ cell keeps two seeds. Width 4 is excluded because the pattern is decodable at the bottleneck at 0.82, against 0.95–0.99 at the wider settings, so it only partly reaches the classifier (Appendix J).

APPENDIX F THE BUILD COST

Build cost by pattern. Table 7 shows that the pattern alone moves the build cost by an order of magnitude, and the order holds on both sets.

Salience and separability. Table 8 compares two input statistics with the build cost. *Salience* is the size of the change the pattern makes to its host, $\mathbb{E}_i \|X_i^{\text{pattern}} - X_i\|_2$ over the training images. *Separability* is how far the pattern sits from what already occupies its window. The *joint* distance is the Mahalanobis distance of the pattern's 108 values from the mean and covariance of the same

Table 7: The build cost depends on the pattern. Backdoor condition, SmallCNN, $n = 7{,}700$; plain set at 400 epochs, shortcut set at 150. Medians over the seeds that pass; the plain faint block rests on three seeds. The watermark is a small random tile repeated over the whole image; its medians come from a separate run at 120 epochs (Appendix I), in which the random block needs 8.5.

pattern	plain	shortcut
solid 6×6 block (PATCH6)	11	7
solid 3×3 block (PATCH3)	23	13
faint 6×6 block (FAINT6)	≈ 80	32
whole-image watermark	—	4–7

6×6 window over the training images. It uses a Ledoit–Wolf covariance, computed in float64; the covariance is too ill-conditioned for float32. The *colour* distance pools every pixel of that window into one distribution of colours, and averages the pattern's 36 per-pixel Mahalanobis distances. The window lies outside the painted corners, so both statistics are identical on the plain and the shortcut set. For scale, a real corner window sits at a median joint distance of 6.79, and a real corner pixel at a colour distance of 1.42. The three patterns sit at 45, 4 and 1.1 times the first. A scrambled crop has the colours of a real crop in another order, so the two share their colour distance by construction, and differ only in arrangement.

Table 8: Separability, not salience. Plain set, SmallCNN, $n = 7{,}700$, 400 epochs. The three input statistics are computed from the images alone, against the 6×6 window the pattern occupies. Each seed draws its own crop, which gives lower counts than the single fixed crop of Figure 6. Thresholds are medians over the seeds that pass (seven, six and six of eight), counting a seed that crosses below the lowest count on the grid at that count. The ± values are standard deviations over four draws of each pattern. The last row is the scale the distances must be read against.

pattern	salience	colour distance	joint distance	k^* median
random block (PATCH6)	3.90 ± 0.07	5.34 ± 0.20	303 ± 14	11.3
scrambled crop (SHUF6)	3.76 ± 1.05	1.35 ± 0.55	27.5 ± 16.3	29.4
real crop (CROP6)	3.78 ± 1.04	1.35 ± 0.55	7.2 ± 2.9	37.8
a real corner, for scale	—	1.42	6.79	—

Host removed, and the pattern made typical. In Table 9, the same patch on a blank field needs 2 examples, against 15 on a real image, read the same way.

Table 9: The build cost against what the pattern competes with, plain set, median thresholds. *Host removed*: the same patch on a real image and on a blank field, read at half the target rate's ceiling in both columns. *Pattern made typical*: a random patch against one cut from a real image at matched salience, read at the zero of the fitted margin. The SmallCNN row is at the plain set's 400-epoch budget. The ResNet-18 and transformer rows were trained for 120 and 600 epochs and are reported as a direction only.

	host removed			pattern made typical		
architecture	on image	on blank	ratio	random	real	ratio
SmallCNN	15	2	0.13	11.3	37.8	3.3
ResNet-18	17.5	1	0.06	16.2	34.0	2.1
ViT	52.5	7	0.13	23.3	65.8	2.8

The two knobs of Figure 6. Table 10 gives the values drawn there. Changing the pattern moves the build cost more than fivefold and leaves the weight within its intervals. The build cost extrapolated from the counter-example sweep matches the measured one at the higher cost too. Extrapolated from $m \geq 30$, the scrambled crop's sweep gives 60.8 $[50.5, 71.0]$ against 61.3 measured. The real crop's gives 64.4 $[57.8, 70.9]$ against 64.4 measured. Along ρ, the slope of the weight is $+0.32$ $[+0.24, +0.38]$.

Table 10: Build cost and weight, moved one knob at a time. Plain set, SmallCNN, $n = 7{,}700$, 400 epochs, seeds 0–7 in every cell; 95% intervals bootstrap seeds. The weight is the slope of k^* on m over $m \in \{0, 30, 60, 100\}$; the build cost is the $m = 0$ cell. The corner predicts the class with probability $(1+\rho)/2$.

knob	value	build cost k_0	weight $\bar{w}$
pattern	random block	11.5 [9.8, 13.2]	0.732 [0.696, 0.774]
	scrambled crop	61.3 [57.2, 64.6]	0.675 [0.639, 0.721]
	real crop	64.4 [61.2, 67.1]	0.698 [0.669, 0.727]
corner ρ	none	11.5 [9.8, 13.2]	0.732 [0.696, 0.773]
	0	13.3 [11.9, 14.6]	0.832 [0.785, 0.879]
	0.3	13.2 [12.0, 14.3]	0.845 [0.791, 0.901]
	0.6	12.0 [10.7, 13.5]	0.966 [0.897, 1.027]
	0.8	10.7 [9.2, 12.2]	1.074 [1.013, 1.134]
	0.95	8.0 [7.3, 8.7]	1.102 [1.055, 1.145]

APPENDIX G THE MISLABEL CONDITION

A class is a bundle of features, each with its own build cost and weight. How their thresholds combine is not modelled (Section 2). The model predicts only the limit, half the class. The intercepts and exact slopes below are finite-size terms of that combination. We report them; we do not interpret them.

The fraction. The class pair is fixed (automobile relabelled deer). The host size N is swept, with n scaled alongside it, eight seeds per size at 400 epochs, of which two to five pass the guards on the plain set. On the plain set the threshold is $k^* = 7.2 + 0.512\,N$ over $N = 150$ to 780 ($R^2 = 0.998$). On the shortcut set it is $k^* = 0.7 + 0.509\,N$ over $N = 80$ to 780 ($R^2 = 0.9999$).

The counter-examples held fixed. Plain set, 400 epochs, eight seeds per N. Every class holds N correctly labelled images, and k further host-class images are added under the target label ($n = 10N + k$). The counter-examples are therefore the host's N correct images at every k. We fit $k^* = c + sN$ over $N \in \{200, 400, 770\}$: least squares on the seed points that pass, with seeds bootstrapped within each N.

pair	intercept c	slope s	$k^*(770)$
automobile → deer	+14.8 [−4.7, +34.2]	0.956 [0.910, 1.002]	749.5 ± 12.5
horse → ship	+13.5 [−3.1, +31.2]	0.975 [0.921, 1.025]	764.0 ± 14.6

The class flips when the flipped images roughly match the correct ones. This run was designed to separate a class's build cost from its weight, treating the class as one pattern. The prediction was a build cost of about 15 reappearing as the intercept, with the slope at the 1.05 implied by the host-size sweep ($k^*(770) \approx 824$). The alternative was a symmetric tie ($c = 0$, $s = 1$). The intercepts sit near 15, but their intervals include zero. The slope 1.05 is excluded on both pairs, and the tie is not rejected. Both designs grow n with N, so any growth of the class's build cost with n also enters each slope. We now read the premise, one build cost and one weight for a class, as the error, and draw nothing from these intercepts. A guard voided crossings where more than 40% of host probes went to neither label, a tie between two weak readings; it removed two of 48 seed-cells. With every guard off, the primary pair gives $c = -6.1$ [−24.9, +12.9], and the replication $c = +35.6$ [+2.2, +72.4].

The margin, carried from the planted patch. Carrying the planted patch's constants to a class, treating the class as one pattern, predicts the class's margin at $N = 300$: $2k^* - N = [\,2k_0 + N(\bar{w} - 1)\,]/(1 + \bar{w})$. Measured on two class pairs, six to eight seeds each:

set	predicted from the planted patch	automobile → deer	horse → ship
plain	−33.3	+4.0 [−1.0, +7.7]	+16.6 [+5.7, +35.9]
shortcut	+25.8	+13.6 [+9.0, +19.7]	+5.4 [−2.3, +13.5]

The prediction lies outside all four intervals, as it should. A planted patch and a class's appearance are different patterns, and the constants belong to the pattern.

APPENDIX H A NON-PARAMETRIC BASELINE: NEAREST NEIGHBOURS ON THE SAME TRAINING SETS

A classifier that memorises its training set and learns nothing else is the natural null for Eq. (1). We build the counter-example training sets exactly as the network sees them: plain set, $n = 7{,}700$, the pinned PATCH6, and m counter-examples spread over the other classes. We classify the same held-out probes with k-nearest neighbours in pixel space (Euclidean, each pixel standardised). The threshold is read with the paper's definition, on three seeds (`knn_baseline.py`, `knn_report.py`).

	build cost (m=0)	slope in m	k_0 against n	held-out acc.
SmallCNN (the paper)	11.6	0.73	$n^{+0.26}$ [a]	—
1-NN	286	0.31	$n^{+0.74}$	0.27
5-NN	379	unstable	$n^{+0.72}$	0.30
25-NN	486 [b]	none for $m \geq 60$	$n^{+0.82}$	0.31

[a] Over $n = 2{,}000$ to $20{,}000$, the range the neighbour classifiers were run on; to $46{,}000$ it is $n^{+0.23}$ (Figure 5). [b] One seed of three crosses.

Three differences, one for each term of the equation. *The build cost* is twenty-five times larger. In pixel space the 6×6 patch is 36 of $1{,}024$ pixels, so a probe's nearest neighbours are chosen by its content. The target is predicted only once poisoned examples are common enough to be retrieved by chance. *The regime* is close to a fraction. The threshold is 3–6% of the training set at every n from $2{,}000$ to $20{,}000$, where the network's build cost stays nearly flat. *The counter-examples* barely register, because they are retrieved no more often than poisoned examples. An idealised vote restricted to the images that carry the pattern would instead cast a plurality among them. Such a vote would give no build cost, and a threshold set by the label split alone. Neither null gives a build cost that stays a count, or a weight that depends on what else the images predict. Both are consistent with the network learning a feature for the pattern.

APPENDIX I OTHER ARCHITECTURES AND SCALES

SmallCNN and a vision transformer on six patterns. Shortcut set, CIFAR-10, $n = 7{,}700$, four seeds per cell. SmallCNN is trained for 120 epochs and the transformer for 600; both fit their training set (whole-set accuracy 0.98–1.00 at every seed's crossing). Three patterns are confined to the bottom-left corner. Three are whole-image watermarks: a small random tile repeated over every pixel, zero-mean, at amplitude α. Thresholds are medians of first grid crossings.

pattern	kind	SmallCNN (per seed)	transformer (per seed)	ratio
PATCH6	corner	8.5 [5, 7, 10, 10]	10 [5, 10, 10, 20]	1.2
PATCH3	corner	15 [15, 15, 15, 20]	17.5 [7, 15, 20, 100]	1.2
FAINT6, $\alpha = 0.3$	corner	50 [30, 50, 50, 50]	62.5 [20, 50, 75, 150]	1.3
watermark, $\alpha = 0.10$, tile 4	whole	6 [5, 5, 7, 10]	20 [15, 20, 20, 30]	3.3
watermark, $\alpha = 0.15$, tile 4	whole	4 [3, 3, 5, 5]	10 [10, 10, 10, 15]	2.5
watermark, $\alpha = 0.15$, tile 8	whole	7 [7, 7, 7, 10]	20 [15, 20, 20, 20]	2.9

The two families agree on the corner patterns. They disagree by 2.5–3.3× on the watermarks, where the per-seed sets do not overlap. The watermarks are SmallCNN's cheapest patterns and the transformer's dearest. The size of the gap is confounded by the budgets (600 epochs against 120). A uniform schedule effect could not, however, act on whole-image patterns alone, so we report the change of ordering, not its size.

ImageNet32. Backdoor condition, PATCH6, three seeds, both sets. ResNet-18 at 120 epochs fits its training set on every seed at its threshold. Its build cost stays a count as the corpus grows:

ResNet-18	$n = 46{,}000$	$n = 150{,}000$
plain set	9.5 [8.2, 10.0, 10.3]	8.1 [6.9, 11.4, 5.9]
shortcut set	14.5 [13.5, 18.8, 11.3]	18.3 [18.7, 20.7, 15.6]

A second ResNet-18 run on the shortcut set, with the same pattern and budget, carries the corpus to $n = 500{,}000$. It fits its training set at every row (whole-set accuracy ≥ 0.99) and passes every guard:

ResNet-18, shortcut set	$n = 46{,}000$	$n = 150{,}000$	$n = 500{,}000$
k^* per seed	15.0, 20.9, 9.4	16.7, 21.0, 11.9	22.8, 27.8, ≤ 20
mean	15.1	16.5	≈ 25

At $n = 500{,}000$ one seed already follows the rule partly at the lowest count tested, 20, so its crossing lies below the grid and is not fitted. On the fitted seeds $k_0 \propto n^{+0.23}$ $[-0.03, +0.44]$, the exponent of the CIFAR-10 sweep. The grid readout, which includes that seed, gives $n^{+0.18}$ $[-0.02, +0.39]$. The $46{,}000$ and $150{,}000$ cells agree with the table above within the spread of seeds.

The vision transformer, trained for 240 epochs on the same data, collapses to a constant predictor. Its whole-set training accuracy is 0.001–0.004, against a chance level of 0.001 for $1{,}000$ classes, on every seed and at both corpus sizes. Its target rate reads 1.0 because it predicts the target for everything. This is why the full-train floor exists (Appendix J). These cells are void, and no threshold is read from them.

APPENDIX J GUARDS

Every stored result passes through one automated checker. Every number in this paper is read from the seeds it passes. A seed passes only if all of the following hold.

1. **Empty control.** With no poisoned examples, the target rate on the probes stays at or below 0.15. Above it, the network predicts the target for some other reason, and no threshold read from that seed means anything.
2. **Poisoned fit.** At the seed's threshold, at least 95% of the poisoned examples are fitted to the target. It is read on those examples alone. They are a few hundred images among thousands, so a poor fit on them can hide inside an overall training accuracy that rounds to one.
3. **Counter-example fit.** The same, for the counter-examples. A counter-example the network has not memorised cannot cancel anything.
4. **Interpolation.** Whole-training-set accuracy is at least 0.95 on every count the threshold is read from. A network that has stopped fitting its training set produces a crossing that is an artefact of the collapse. Rows above the threshold that fail are skipped by the readout, not read. This guard is why plain-set runs shorter than 400 epochs are not used.
5. **Full-train floor.** A network collapsed to predict the target everywhere passes the first two guards: its poisoned fit reads one and its empty control reads zero. So whole-set accuracy must also be far above chance. This is what voids the vision transformer on ImageNet32.
6. **On-flip fit** (mislabel condition). The flipped examples are fitted before any threshold is read.
7. **Dynamic range.** The true-label accuracy must fall by at least 0.3 over the sweep. A threshold read against a competitor that does not move is set by the readout.
8. **Pattern legibility** (capacity sweep). The pattern must be decodable at the bottleneck. It reaches the bottleneck only partly at width 4, which is therefore excluded.

Sensitivity to the guards. Whether a seed passes depends on how it trains, so the guards could bias the thresholds they keep. Table 11 recomputes the checks of Section 4 under four rules. The first is the guards as above. The next two lower every 0.95 floor to 0.90, or raise it to 0.99: poisoned fit, counter-example fit, interpolation, and the readout's row filter. The last uses no guards at all:

every seed of every file, and every row in the readout. No conclusion changes. The measured build cost always falls inside the interval of the extrapolated one. Every held-out cell stays inside its 95% interval, except the shortcut $m = 100$ cell reported in Section 4. That cell returns inside under every other rule. Parity keeps its sign on both sets. With no guards, one shortcut seed of 18 falls below parity at $m \geq 60$.

Table 11: Guard sensitivity. Sigmoid readout throughout. k_0: backdoor build cost (seeds that pass). Extrap. k_0: the build cost extrapolated from the counter-example sweep fitted without its $m = 0$ cell, 95% bootstrap interval. LOO: each cell predicted from a weight fitted on the others (χ^2, cells inside 95%). Extrap.: weight fitted on $m \leq 30$, cells $m = 60$–150 predicted. Sweep parity: plain, seeds with $k^* < m$ at $m \geq 60$; shortcut, seeds with $k^* > m$ at every m. $k{=}m$ run: seeds whose target rate beats the true-label accuracy at $k = m$, for $m = 10, 30, 50$; the main text quotes this run with no guards. The first row of each set recomputes the main-text numbers with this table's bootstrap draws, which can move the last digit. Script: `guard_sensitivity.py`.

set	rule	k_0	extrap. k_0	$\bar{w}$	LOO	extrap.	sweep parity	$k{=}m$ run
PLAIN	guards (0.95)	11.6 (5)	10.5 [8.2, 12.9]	0.731	3.6, 5/5	1.0, 3/3	31/31	0/5, 1/4, 5/6
	floors 0.90	11.6 (5)	10.7 [8.3, 13.2]	0.730	2.2, 5/5	0.3, 3/3	36/37	0/6, 1/4, 5/7
	floors 0.99	11.3 (3)	11.6 [9.3, 14.0]	0.715	0.5, 5/5	0.3, 3/3	18/18	0/3, 1/3, 4/5
	no guards	11.6 (5)	11.0 [8.5, 13.6]	0.747	0.4, 5/5	0.1, 3/3	45/48	0/8, 3/8, 5/8
SHORTCUT	guards (0.95)	7.0 (4)	6.5 [3.4, 10.2]	1.137	8.1, 4/5	2.4, 3/3	26/26	0/6, 0/5, 1/6
	floors 0.90	7.0 (4)	7.2 [3.8, 11.0]	1.120	3.7, 5/5	3.0, 3/3	28/28	0/7, 0/5, 1/6
	floors 0.99	6.7 (3)	7.8 [4.9, 11.2]	1.119	5.5, 5/5	2.9, 3/3	22/22	0/6, 0/4, 1/6
	no guards	7.3 (5)	8.8 [4.6, 13.7]	1.084	2.0, 5/5	2.9, 3/3	29/30	0/8, 0/8, 1/8

Sensitivity to the readout. k^* is the zero of a logistic fitted to the margin over every count. Two simpler readouts bracket it: the first interpolated count at which the target rate beats the true-label accuracy, and the last such crossing that does not come back. They agree for seeds that cross once. About half the plain seeds (46%) and a quarter of the shortcut seeds (25%) cross more than once; for those, the band between the two averages 0.35 and 0.33 of k^*. Table 12 repeats the checks of Section 4 under each readout, with the guards on. The weight keeps its side of one on both sets, and the build-cost check passes everywhere. The first crossing loses one held-out plain cell and puts two shortcut seeds below parity. The last crossing leaves 9 of 30 plain seeds above parity at $m \geq 60$. The first-crossing losses come from grid density: a first crossing is a minimum over noisy comparisons, so cells with more counts near k^* read low. Taking the first and last crossings on the raw grid instead gives plain weights of 0.694 and 0.827, and shortcut weights of 1.135 and 1.286.

Table 12: Readout sensitivity. Guards on throughout. k_0: backdoor build cost. Extrap. k_0: the build cost extrapolated from the counter-example sweep fitted without its $m = 0$ cell, 95% bootstrap interval. LOO: each cell predicted from a weight fitted on the others (χ^2, cells inside 95%). Extrap.: weight fitted on $m \leq 30$, cells $m = 60$–150 predicted. Parity: plain, seeds with $k^* < m$ at $m \geq 60$; shortcut, seeds with $k^* > m$ at every m. Mixed: largest deviation of a mixed cell (Table 1), in standard errors. Scripts: `guard_sensitivity.py`, `table5_additivity.py`, with `-kstar sigmoid`, `interp`, `last_interp`.

set	readout	k_0	extrap. k_0	$\bar{w}$	LOO	extrap.	parity	mixed
PLAIN	fitted zero	11.6	10.5 [8.2, 12.9]	0.731	3.6, 5/5	1.0, 3/3	31/31	2.0
	first crossing	12.4	10.2 [7.2, 12.9]	0.675	7.1, 4/5	2.7, 3/3	31/31	1.5
	last crossing	12.4	13.5 [10.0, 17.6]	0.794	1.2, 5/5	0.5, 3/3	21/30	1.7
SHORTCUT	fitted zero	7.0	6.5 [3.4, 10.2]	1.137	8.1, 4/5	2.4, 3/3	26/26	2.3
	first crossing	7.0	7.3 [2.4, 12.8]	1.104	10.2, 4/5	4.9, 3/3	24/26	2.0
	last crossing	7.0	7.7 [0.4, 15.9]	1.247	1.4, 5/5	0.7, 3/3	26/26	1.4