CORRESPONDENCE OVER CONTENT: WHAT MAKES TASK INFORMATION READABLE IN NEURAL NETWORK WEIGHTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Weight space methods read trained networks to retrieve, compare and edit them. We ask what makes the information in a network's weights readable, and what it costs to read. Studying collections of networks that differ only in how a fixed set of images is labelled, so that the task is the only variable, we find that correspondence between models governs not what a reader can recover but what recovering it costs. When fixed sparse wiring pins each unit to the same inputs in every network, a linear probe identifies the task from a few hundred first layer weights or fewer, with no net growth over a sixteenfold range of width. Dense networks hold the same information in linearly readable form, but only in aggregates over their hidden units: the cheapest reader we found must read every unit, and its cost grows with the layer, with an exponent between 0.7 and 0.8. Matching units to a template, on a few unlabelled images or on the weights alone, brings the dense budget back to a few hundred coordinates, so the premium for missing correspondence itself grows with the network, from threefold to seventeenfold across our widths. In the hidden to hidden layer, removing the permutation symmetry with fixed wiring is not enough: its coordinates stay unreadable, so removing a symmetry is not the same as supplying correspondence. Along the way we document how standard readings mislead: per model scale alone identifies the training dataset on a public model zoo, and an alignment step can itself carry the task.

1 INTRODUCTION

A trained network's weights are a record of what it learned, and a growing body of work reads that record: to predict how a model will behave, to retrieve one from a collection, to merge or edit it, or to generate new ones. The question these methods share is how much can be recovered from parameters. The question this paper asks is what makes recovery possible, and what it costs.

The two questions come apart because of correspondence. A network's hidden units are interchangeable, and its initialisation decides which unit ends up doing what, so two networks trained on the same data place the same computation at different coordinates. A reader that compares parameter vectors across networks is therefore reading a code whose alphabet is reshuffled from one model to the next. The field's usual responses are to remove the reshuffling by aligning models, or to absorb it by building encoders that are invariant to it. Both treat it as an obstacle. Neither measures its price.

Measuring it requires collections of networks in which the task is the only thing that varies. Public model zoos do not have that property: they change the dataset, and with it the input statistics and the number of optimiser steps, so a statistic that separates their models may be reporting any of these. We therefore train our own, holding the images, the schedule and the optimiser fixed and varying only how the images are labelled. Each labelling is a task on identical pixels; each is trained from many initialisations; and every measurement holds initialisations out, so a reader always meets networks it has not seen. We train two families on this protocol, one with sparse wiring fixed before training and shared by every network, one dense, and we measure how many coordinates a reader must touch to identify the task in each (Figure 1).

The answer comes in two parts. Both families hold the task, and both hold it in a form a linear reader can use. With fixed wiring, where a unit reads the same inputs in every network, a probe identifies

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.

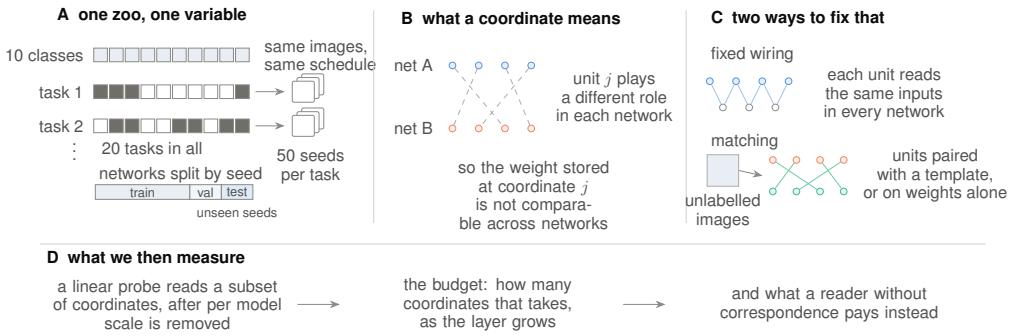


Figure 1: The experiment. A: networks differing only in how a fixed set of images is labelled, many initialisations per labelling. B: the same unit index holds a different role in each network, so a coordinate is not comparable across them. C: two ways to make coordinates comparable, one supplied by the wiring, one bought with a matching on unlabelled inputs or on the weights. D: what we then measure.

the task from a few hundred first layer weights or fewer, with no net growth as the layer widens sixteenfold. Trained dense, the same networks hide nothing: averages over their hidden units, taken pixel by pixel, identify the task perfectly. But those averages are only available to a reader that reads every unit, so the cheapest reading we found grows with the layer. Matching the hidden units against a template, on a small batch of unlabelled inputs or on the weights alone, restores a budget of a few hundred coordinates. What separates the cases is not how much information the weights carry but what a reader must pay to reach it, and the gap between paying for correspondence and going without it widens with size.

This locates the price precisely. The input side of the first layer is a fixed axis, the one on which equivariant weight space architectures already treat input neurons as distinguished (Navon et al., 2023; Zhou et al., 2023); readers that ignore unit order can use it, at a cost that grows with the number of units. A layer whose both sides are hidden has no such axis. There, removing the permutation symmetry with fixed wiring, the device recent work uses to make networks easier for metanetworks to read (Lim et al., 2024), does not make coordinates comparable: the hidden to hidden layer of our wired networks stays unreadable. Removing a symmetry and supplying correspondence are different things. The results also make a warning necessary. Several readings we tried recover the task for reasons unrelated to the structure they appear to read: the scale of a layer, the zero pattern of a sparse one, and, in a matched frame, the matching itself. Each produced a result we believed before a control removed it.

Contributions.

- Zoos of networks that differ only in their labelling, with a null arm that measures each instrument's floor on real data, and a reading protocol that removes per model scale before it measures structure.
- A measurement of what reading costs. With fixed wiring or after matching, a few hundred coordinates suffice with no net growth over a sixteenfold width range; without correspondence, the cheapest reader we found aggregates over every hidden unit and its cost grows with the layer, so the premium for missing correspondence grows with the network.
- Evidence that content does not limit a reader: five readers that need no correspondence recover the dense first layer perfectly, a matching needs no images, and in the hidden to hidden layer fixed wiring removes the symmetry without supplying correspondence.
- A set of controls for weight space measurement, each of which changed a result of ours, and a public zoo on which a handful of scalars already saturates the task, which is a statement about that benchmark rather than about any method evaluated on it.

2 ZOOS IN WHICH THE TASK IS THE ONLY VARIABLE

Asking where a network's task is written in its weights requires collections of networks that differ in their task and in nothing else. Public model zoos do not meet that condition: they vary the training dataset, and with it the input distribution, the number of images and therefore the number of optimiser steps. A weight statistic that separates such models may be reporting the task, the pixel statistics of the images, or simply how long each network trained.

We remove those alternatives by construction. Every network in our zoos sees the same images for the same number of epochs, with the same optimiser and the same hyperparameters. Only the labelling differs: we draw random partitions of the CIFAR-10 classes into two groups, and each partition is a task defined on identical pixels. Each task is trained from the same fifty initialisations, and every measurement splits models by seed, so a reader is always tested on initialisations it has never seen. Appendix A gives the architecture, hyperparameters and zoo sizes.

Two families are trained on exactly this protocol. In the *masked* family all networks share one sparse wiring pattern, drawn once per architecture, so a hidden unit reads the same inputs in every network. In the *dense* family the same networks are trained with no wiring constraint. The two families let us ask what the wiring buys with everything else held fixed. Both span a sixteenfold range of width.

Why fixed sparse wiring. The masked family is an instrument, not a claim about sparse networks. In a densely connected layer, permuting the hidden units and their outgoing weights leaves the function unchanged, which is exactly why a coordinate in one network need not mean what it means in another (Entezari et al., 2022; Ainsworth et al., 2023). Under a fixed mask whose rows have distinct supports, that permutation no longer preserves the architecture: a permuted unit would have to read different inputs, so the symmetry is absent by construction rather than removed after the fact. Networks trained from fixed random sparse connectivity, chosen before training and never updated, reach accuracies close to their dense counterparts (Liu et al., 2022; Prabhu et al., 2018). Lim et al. (2024) break the same symmetry with fixed entries unique to each row and find that metanetworks read the resulting networks more easily. We use the device to measure what it buys a reader, and find that it buys correspondence only where a layer's inputs are themselves fixed (Section 8).

Alongside each family we train a *null* arm: the same protocol with a single labelling shared by all groups, and the groups formed from disjoint blocks of seeds. The task is constant there by construction, so whatever a measurement reports on the null arm is its floor on real data. The masked null arm spans all five widths and the dense one is trained at width 128; every number below with a null counterpart is reported beside it.

3 MEASURING WHAT CAN BE READ

Readers. A reader maps a network's weights to a prediction of its task. Our main reader is a linear probe on coordinates: a subset S of a layer's weights, standardised and projected onto principal components fitted on training models only, with the regularisation strength chosen on validation models and the probe scored once on test models. Models are split by seed, and every probe is paired with the same fit on permuted labels. We use a linear probe deliberately: the question is how much of a network must be read to recover its task, not how well a large model fits a small zoo. Two further kinds of reader need no correspondence at all. *Invariant readers* use statistics that do not depend on the order of hidden units, such as per pixel quantiles and moments over units, or singular values. The *aggregate reader* averages a layer over u of its units, pixel by pixel, and probes m of those pixel averages; it reads mu coordinates.

Scale is removed first. Tasks differ in difficulty, difficulty shapes the loss trajectory, and the trajectory sets how far weights travel, so a few per layer summary statistics already recover much of the task (Table 4); on a public zoo four numbers name the training dataset almost perfectly (Section 9). That footprint is real but it is not structure. Every decode result below is measured after standardising each network's layer over its live weights, so that scale and offset carry nothing and what remains is the shape of the weight pattern.

Reading budget. Sufficiency alone is weak, since reading a whole network locates nothing. For a reader of cost n, let $a(n)$ be its test accuracy. The budget is

$$n^* = \min\{\, n : a(n) \geq \tau \,\}, \qquad \tau = 0.95, \tag{1}$$

interpolated in $\log n$ on a geometric ladder of n. The bar τ is absolute and common to every reader and every width. Every first layer reader we report reaches 1.0 when given enough coordinates, except the coordinate probe on dense layers read directly, which never reaches the bar and so has an infinite budget. For the coordinate probe, n is the size of a ranked or of a uniformly random subset; for the aggregate reader, n^* is the cheapest cell (m, u) of a grid that clears the bar. We fit $\log n^*$ against the logarithm of the number of live weights the layer really has, N, and repeat the whole measurement over ten seed splits; we report per width medians with their range and the fitted exponent with its interval. An exponent whose interval excludes one is sublinear; we do not read constancy from an interval that merely contains zero. Appendix B states the estimator and the design errors its self test caught.

4 Correspondence between networks

A coordinate in one network and the same coordinate in another are comparable only if the units they connect play comparable roles. Nothing in a standard parameterisation guarantees this. Hidden units are interchangeable and the initialisation decides which unit does what, so two networks trained on the same task place the same computation in different coordinates. We call this the correspondence problem. Our two families address it in two ways.

Fixed sparse wiring addresses it at training time: a unit that can only read a fixed subset of the inputs is constrained in the roles it can take, so networks trained from different seeds place similar computations at similar coordinates.

Matching addresses it at reading time. We record each network's hidden activations on a small batch of unlabelled images, standardise each unit's profile, and pair units between a network and a template by maximising the total correlation of matched pairs, an assignment problem solved exactly (Li et al., 2016; Ainsworth et al., 2023). For the first layer we also match on the weights alone, pairing rows by correlation with no images at all. Templates are built from training models only and then frozen; no labels enter, and no test model shapes the frame in which it is later read. Matching turns correspondence into something that can be bought, and the number of images used is the price, so we sweep it. We report each matching's quality on images it did not use, and we write the aligned networks back out in the original format, so the budget of Section 3 runs unchanged in the aligned frame.

5 What would fool us

Most of the machinery above exists to catch failures we met. Each of the following changed a number we had already believed, and each is reported with the experiment it protects. *Scale*: summary statistics decode the task on their own, so we standardise, and we quote the statistics baseline. *Structural zeros*: in a masked network most entries are zero by construction, and standardising with them included copies each model's scale across thousands of fixed positions, so we standardise live entries only. *Information carried by the alignment*: a matching is computed from a network's own behaviour and is therefore a function of its task, so reading weights in a matched frame can recover the task even when those weights carry nothing task specific; we test this by applying each network's matching to a fixed task free matrix, by probing the matchings themselves, and by probing each half of the alignment alone. *Distance travelled*: how far a layer moves from its initialisation changes with width and differs between families, and a layer dominated by its initialisation is harder to read for reasons unrelated to correspondence, so we report it beside every width trend (Appendix F.4). *The pipeline*: every control is run on the null arm, since a template, a split and an export path are all chances to manufacture structure. Appendix C gives each control in full, including one that failed its own validity check and what we concluded instead.

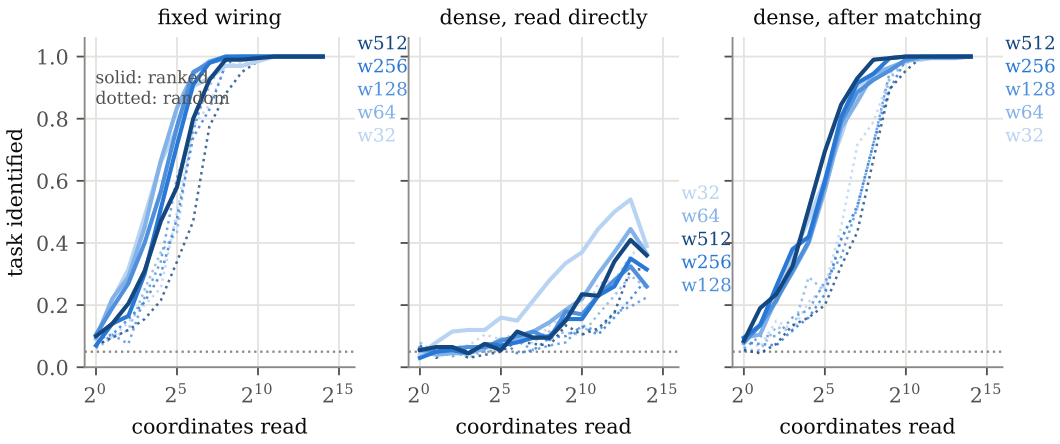


Figure 2: What a coordinate reader gets for what it reads. Task identification against the number of first layer coordinates read, one line per width, ranked coordinates solid and random dotted. Left: with fixed wiring the curves for all five widths lie close together. Centre: the same networks trained without wiring constraints, read coordinate by coordinate, never approach the ceiling. Right: after matching their hidden units on a small batch of unlabelled images, the curves return to the left panel's shape. Chance is the dotted horizontal line.

6 Under fixed wiring the task is readable, and cheaply

In the masked family, a linear probe recovers the task from the first layer perfectly at every width, and the second layer is at chance against its permutation null. This holds after each model's layer has been standardised, so it is not the scale footprint of Table 4, and the null arm never rises above 0.09 at any width, against a chance level of 0.05.

The budget is the interesting quantity. The left panel of Figure 2 shows what a reader recovers as a function of how many coordinates it reads, at every width, and Figure 3 shows the budget itself. Over ten seed splits, the median budget for ranked coordinates is 219, 87, 80, 103 and 200 from the narrowest width to the widest, and for random coordinates 373, 164, 168, 226 and 419. Across a sixteenfold range the budget shows no net growth: the fitted exponent is 0.00 for ranked and 0.08 for random coordinates, with intervals that exclude proportionality. It is not constant either. The curve is shaped like a U, lowest at width 128, and rises by about two and a half times between widths 128 and 512. The rise is not a property of one wiring pattern: two further masks drawn at width 128 put the ranked budget between 64 and 83. Because the count stays in the hundreds while the layer grows, the fraction of a network a reader must touch falls from 2.2% to 0.13%.

Selection helps, and supervised selection helps most. Ranking coordinates by a supervised statistic roughly halves the budget of a random subset at every width, while the label free rankings we tested sit within noise of random in this family (Appendix F.5).

7 Dense networks hold the task, but reading it costs more with size

Trained without masks and read coordinate by coordinate, the same first layer never reaches the bar: the probe tops out between 0.24 and 0.40 however much of it is read (Figure 2, centre), with or without standardising.

It is not hidden at all. Table 1 reads the same dense layers with readers that need no correspondence. Per pixel quantiles and moments over the hidden units, a summary that cannot depend on unit order, identify the task perfectly at every width, and so does the smallest such summary, the per pixel mean and mean square. The linear half alone suffices: per pixel means over units, a linear function of the weights, reach 1.000 at widths 128 and 512. The dense first layer therefore carries its task in linearly readable form. What the coordinate probe lacks is not access to the information but enough of each

Table 1: Readers of the dense first layer that need no correspondence. Test accuracy at full reading, relabelled zoo, chance 0.05. Invariant readers do not depend on the order of hidden units; weight matching pairs each network's rows with a template by correlation, using no images. On the null arm at width 128 every reader stays at or below 0.06.

Reader	32	64	128	256	512
Coordinates, $20{,}000$ stored	0.375	0.375	0.275	0.260	0.370
Per pixel quantiles and moments	1.000	1.000	1.000	1.000	1.000
Per pixel mean and mean square	1.000	1.000	1.000	1.000	1.000
Singular values, top 32	0.215	0.315	0.360	0.475	0.485
Weight matching, then coordinates	1.000	1.000	1.000	1.000	1.000

pixel's units: the $20{,}000$ coordinates it reads give each pixel only a handful of them, and the task lives in their average.

That average has a price, and the aggregate reader measures it. Averaging u units at m pixels costs mu coordinates, and at every width and on every seed split the cheapest cell that clears the bar uses every unit, with 32 to 64 pixels. Pixels can be subsampled freely; units cannot. The budget therefore grows with the layer, from $2{,}048$ coordinates at width 32 to $16{,}384$ at width 512, with a fitted exponent of 0.70 to 0.80 across three seed splits, every interval excluding zero (Figure 3). The null arm never clears the bar at any cell. The pattern is forced for linear readers: a linear functional of a pixel's weights that ignores unit order satisfies $a^\top P v = a^\top v$ for every permutation P, so $a = c\,\mathbf{1}$, and it must read every unit of every pixel it uses. The aggregate reader attains this bound; only a nonlinear reader could do better without correspondence.

Correspondence brings the cost back down. Matching hidden units to a template restores the coordinate probe at every width (Figure 2, right), and it does so whether the matching is computed from a batch of unlabelled images or from the weights alone (Table 1); for the first layer the price of correspondence is therefore not even a forward pass. In the matched frame the ranked budget falls from 431 to 161 across widths and the random one moves from 619 to 967, an exponent of 0.15 whose interval contains zero. Set against the aggregate reader, random subsets for both, the premium for missing correspondence is 3.3, 6.1, 8.0, 10.7 and 16.9 times across widths, and grows as $N^{0.55}$ with an interval of 0.41 to 0.70. Correspondence does not change what a reader can recover from this layer. It changes how the cost of recovering it scales.

8 The deeper layer, where no axis is fixed

The input side of the first layer is indexed by pixels, which mean the same thing in every network; the readers of Table 1 rely on that. The hidden to hidden layer has no such axis: both of its sides are indexed by units. Here fixed wiring removes the permutation symmetry and still leaves the coordinates unreadable, at chance at every width, while a permutation invariant summary of the same layer reaches about seven times chance. A masked unit in this layer is pinned to fixed inputs, but those inputs are units whose roles the seed decides, so the pinning buys nothing. Removing a symmetry, the property that makes asymmetric networks easier for metanetworks (Lim et al., 2024), is not the same as supplying correspondence.

In the dense family, matching both sides makes this layer readable, and increasingly so with width (Table 2); its curve against the size of the matching batch rises and saturates, and where it saturates depends on width (Figure 4). The attribution needs care. A matching is computed from a network's behaviour and is therefore a function of its task, so weights read in a matched frame can carry the task even when the weights themselves carry nothing specific to it (Appendix C). In the dense family the matchings alone carry nothing and neither half of the alignment works without the other, but the control that would separate the layer's values from its frame failed its own validity check (Appendix C). A second confound is independent of the frame: the share of this layer set by training rather than by initialisation grows over these widths, its distance from initialisation rising from 0.60 to 1.06 times its initial norm (Appendix F.4). The width trend in the dense second layer, matched or summarised, is therefore real but not yet attributable to correspondence, and we do not attribute it.

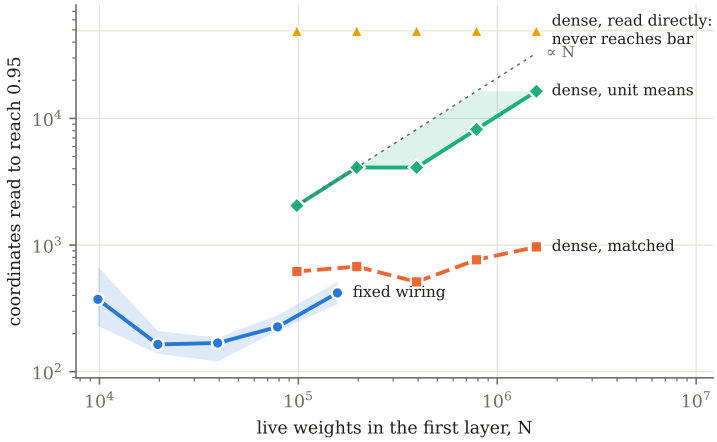


Figure 3: Coordinates a reader needs to identify the task at accuracy 0.95, against the number of live weights in the first layer. Fixed wiring and the matched dense frame read random coordinates; the dense aggregate reader averages every unit at randomly chosen pixels; the dense coordinate probe never reaches the bar. Bands span seed splits (ten for fixed wiring, three for the aggregate reader); the matched frame uses the split its template was built on. At equal width, the masked layer has a tenth of the dense layer's live weights, which shifts its curve left. Budgets are resolved to a factor of two. The dotted line has slope one.

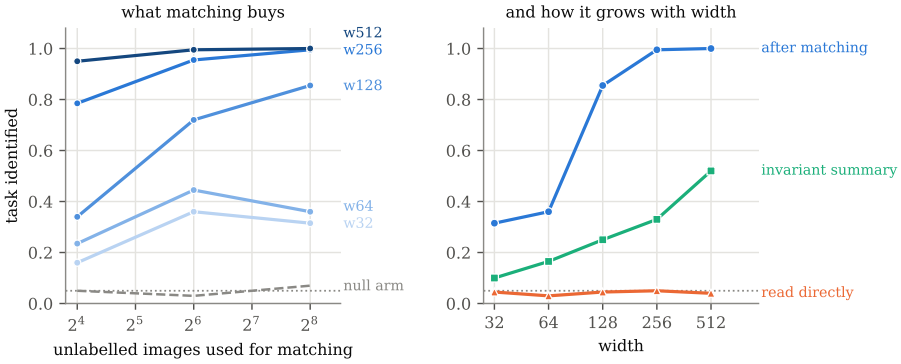


Figure 4: The second layer of dense networks. Left: task identification against the number of unlabelled images used to match hidden units, one line per width, with the null arm flat near chance. Right: the same layer at the largest matching batch, against width, beside two readings that use no matching. Readable content grows with width in the matching free summary as well, so no property of the frame explains it alone; the share of the layer set by training grows over the same range (Appendix F.4). The first layer is not shown: it is at ceiling at the smallest matching batch, and with no images at all.

9 A PUBLIC ZOO THAT CAN NO LONGER DECIDE

Finally we apply the same pipeline to a public zoo of 2,000 small convolutional networks, one hundred seeds on each of twenty image datasets, released with WeightCLIP (**?**); we keep one checkpoint per run and split by run. Permutation invariant features identify the training dataset perfectly. So do four numbers: the per layer norms reach 0.97 (Table 3), and every layer on its own is above 0.95 even after each layer is rescaled to unit norm. A benchmark on which four scalars leave three points of headroom cannot compare weight representations, whatever it reports about any particular method. The finding is about the evaluation rather than about the networks, and it is the reason the controlled zoos carry the rest of this paper.

Table 2: Matching quality and what it buys, dense family, one matching batch per width. Quality is the mean correlation of matched units on images not used for matching, or of matched rows for the weight matching. The permutation invariant summary of the second layer involves no matching and cannot carry information through one. Budgets use ranked coordinates at accuracy 0.95.

Width	32	64	128	256	512
Match quality, first layer, images	0.59	0.65	0.70	0.76	0.84
Match quality, first layer, weights	0.26	0.28	0.32	0.37	0.40
Match quality, second layer	0.49	0.51	0.50	0.44	0.39
Budget, first layer, matched	431	304	456	274	161
Second layer, matched	0.32	0.36	0.86	0.94	0.98
Second layer, invariant summary	0.10	0.17	0.25	0.33	0.52

Table 3: A public zoo of convolutional networks, identifying the training dataset among twenty; chance is 0.05 and the permutation null is 0.035. The classifier head is dropped so that datasets with different class counts share one architecture; features are permutation invariant with the last kept layer treated as hidden; the regularisation strength is chosen on validation runs. With every layer rescaled to unit norm first, the first layer alone reaches 0.98 and the remaining layers together 0.98.

Features	**Dimension**	**Probe**
Four per layer norms	4	0.973
Permutation invariant features, all layers	1108	1.000
First layer only	572	0.998
Everything except the first layer	536	0.985

10 Related work

Reading trained weights. Properties of a trained network can be predicted from its parameters: accuracy and hyperparameters from simple per layer statistics (Unterthiner et al., 2021), the training dataset from a classifier over weights (Eilertsen et al., 2020), and a range of attributes from learned encodings of model zoos (Schürholt et al., 2022a;b; 2024), or from the network's responses to learned probe inputs (Kahana et al., 2025). Recent work audits this literature, showing that zoo composition matters more than zoo size (Falk et al., 2025) and that generative models of weights largely reproduce or interpolate their training checkpoints (Zeng et al., 2025). Our statistics baseline is in that spirit and reaches the same conclusion on a public zoo; we then ask not how well weights can be read but what reading them costs.

Permutation symmetry and alignment. Hidden units can be permuted without changing a network's function. Independently trained solutions become linearly connected after alignment (Entezari et al., 2022; Ainsworth et al., 2023; Jordan et al., 2023); alignment can be posed as matching or optimal transport (Singh & Jaggi, 2023; Tatro et al., 2020) and learned (Navon et al., 2024); and units of different networks can be paired by the correlation of their activations (Li et al., 2016), which is our matching step. A parallel line builds the symmetry into the architecture that reads weights (Navon et al., 2023; Zhou et al., 2023; 2024; Kofinas et al., 2024; Lim et al., 2023); there, input and output neurons are not permuted, which is the fixed axis our invariant readers use. Closest to our masked family, Lim et al. (2024) remove the symmetry at training time and find asymmetric networks easier to read. These lines treat symmetry as an obstacle to remove or absorb. We measure what a reader pays when it does neither, and find that removing the symmetry supplies correspondence only where a layer's inputs are fixed.

Correspondence between representations. That two networks solving the same task need not place the same computation in the same units is familiar from representation analysis, through similarity measures (Kornblith et al., 2019) and through stitching (Lenc & Vedaldi, 2015; Bansal et al., 2021). Our correspondence problem is the weight space form of the same question.

Fixed sparse wiring. Sparse masks fixed before training are studied for efficiency and for what they say about initialisation (Frankle & Carbin, 2019; Ramanujan et al., 2020; Malach et al., 2020), and random ones train close to dense accuracy (Liu et al., 2022; Prabhu et al., 2018). We use them for an unrelated reason: a unit constrained to fixed inputs keeps a comparable role across networks. That the first layer results hold in dense networks once a matching is applied keeps them from being a property of sparse models.

Information for a bounded reader. Probing accuracy alone does not say how much of a task a representation carries, which motivates description length (Blier & Ollivier, 2018; Voita & Titov, 2020). Usable information is relative to a class of readers (Xu et al., 2020; Finzi et al., 2026), and computation such as a matching can create it; our budget is a measurement of that kind, with the reader class fixed and its cost counted in coordinates.

Width and what training moves. How far each layer moves from its initialisation depends on width, parameterisation and optimiser (Yang & Hu, 2022; Everett et al., 2024). We report that distance beside every width trend, because a layer dominated by its initialisation is harder to read for reasons that have nothing to do with correspondence.

11 Discussion

The results say that a network's first layer carries its task in a form any linear reader can use, and that correspondence decides what reading it costs: a few hundred coordinates when the wiring or a matching supplies it, and a cost that grows with the layer when nothing does. Three consequences follow for weight space methods.

First, encoders that tokenise coordinates work in the regime where correspondence is unpaid, and there the cost of reading grows with the model. Paying for it is cheap: a matching on the weights alone, with no images, labels or gradient steps, makes the first layer readable from a few hundred coordinates at every width we tried. Whether to amortise that matching into an encoder or perform it as preprocessing is an empirical question this paper does not settle, but it is worth putting before the architectural one.

Second, where a layer has a fixed axis, readers that ignore unit order recover the task without any correspondence, which is the axis equivariant architectures already exploit; where it has none, as between two hidden layers, neither those readers nor fixed wiring recover it, and only a matching does, with an attribution we cannot yet settle. The hard part of reading weights is in the layers with no fixed axis.

Third, knowing where to look helps but is not yet available to a reader that does not know the task: supervised rankings roughly halve the budget, label free ones track random under fixed wiring and beat it only in the matched frame (Appendix F.5). And which coordinates are easiest to read makes no measurable difference to adaptation.

Limitations. Our networks are small two layer perceptrons on one image dataset with twenty binary relabellings, trained with one optimiser and schedule. At this task count every first layer reader reaches the ceiling, so readers are compared by budget rather than by accuracy. The aggregate reader is the cheapest we found, so its cost is an upper bound; the budgets are resolved to a factor of two, the matched frame rests on the one seed split its template was built on, and the budget under fixed wiring is not monotone in width. The two families differ in more than correspondence: under the same default initialisation, scaled by the dense fan in, masked first layers travel 1.6 to 1.9 times as far from their initialisation as dense ones, and in both families the distance falls with width (Appendix F.4). In the layer whose both ends are hidden, we cannot separate what the aligned weights carry from what the matching carries, and the width trend there coincides with a growing share of the layer set by training. Finally, the public zoo we examine is saturated by summary statistics, so it tells us about that benchmark rather than about convolutional networks in general; whether the cost of reading without correspondence grows as it does here in convolutional and attention based zoos, where a first layer has far fewer fixed slots, is the obvious next experiment.

REPRODUCIBILITY STATEMENT

Every experiment in this paper runs from the scripts in the supplementary material. Section 2 and Appendix A give the architectures, hyperparameters, zoo sizes and the seed based splits; the zoos themselves are generated by one script from a single command line, and the figures and tables are produced from the analysis scripts' own machine readable output rather than from transcribed numbers, so a figure cannot drift from the run that produced it. Each measurement script carries a self test that plants a known ground truth, including cases designed to be recovered as negative, and Appendix B and Appendix C report the design errors those self tests caught before the measurements were trusted. Budgets are repeated over seed splits and, under fixed wiring, over three independently drawn masks at one width. The registered predictions made before each experiment, and their outcomes including the ones that failed, are listed in Appendix E.

AI USE STATEMENT

Generative AI tools were used throughout this work, in the following ways. Code: the experiment, analysis and figure scripts were written with the assistance of an AI coding assistant; every script carries a self test that plants a known ground truth, and several design errors introduced during that process were caught by those tests and by the controls reported in Appendix C before any result was trusted. Analysis and interpretation: hypotheses, decision rules and predictions were developed in dialogue with an AI assistant and registered in writing before each experiment was run, with outcomes recorded whether or not they matched (Appendix E); several claims made during that process were retracted when the evidence arrived, and the paper reports the corrected versions. Writing: drafts of the text were produced with AI assistance and revised by the authors. Literature: candidate related work was suggested with AI assistance and every citation was verified by the authors against the primary source; suggestions that could not be verified were removed. The authors reviewed all AI assisted work and take responsibility for the final content of this paper, including its text, claims and artifacts.

REFERENCES

Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git Re-Basin: Merging Models modulo Permutation Symmetries, March 2023. URL http://arxiv.org/abs/2209.04836. arXiv:2209.04836 [cs.LG].

Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting Model Stitching to Compare Neural Representations, June 2021. URL http://arxiv.org/abs/2106.07682. arXiv:2106.07682 [cs.LG].

Léonard Blier and Yann Ollivier. The Description Length of Deep Learning Models, November 2018. URL http://arxiv.org/abs/1802.07044. arXiv:1802.07044 [cs.LG].

Gabriel Eilertsen, Daniel Jönsson, Timo Ropinski, Jonas Unger, and Anders Ynnerman. Classifying the classifier: dissecting the weight space of neural networks, February 2020. URL http://arxiv.org/abs/2002.05688. arXiv:2002.05688 [cs.CV].

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The Role of Permutation Invariance in Linear Mode Connectivity of Neural Networks, July 2022. URL http://arxiv.org/abs/2110.06296. arXiv:2110.06296 [cs.LG].

Katie Everett, Lechao Xiao, Mitchell Wortsman, Alexander A. Alemi, Roman Novak, Peter J. Liu, Izzeddin Gur, Jascha Sohl-Dickstein, Leslie Pack Kaelbling, Jaehoon Lee, and Jeffrey Pennington. Scaling Exponents Across Parameterizations and Optimizers, July 2024. URL http://arxiv.org/abs/2407.05872. arXiv:2407.05872 [cs.LG].

Damian Falk, Konstantin Schürholt, and Damian Borth. The Impact of Model Zoo Size and Composition on Weight Space Learning, April 2025. URL http://arxiv.org/abs/2504.10141. arXiv:2504.10141 [cs.LG].

Marc Finzi, Shikai Qiu, Yiding Jiang, Pavel Izmailov, J. Zico Kolter, and Andrew Gordon Wilson. From Entropy to Epiplexity: Rethinking Information for Computationally Bounded Intelligence, March 2026. URL http://arxiv.org/abs/2601.03220. arXiv:2601.03220 [cs.LG].

Jonathan Frankle and Michael Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks, March 2019. URL http://arxiv.org/abs/1803.03635. arXiv:1803.03635 [cs.LG].

Keller Jordan, Hanie Sedghi, Olga Saukh, Rahim Entezari, and Behnam Neyshabur. REPAIR: REnormalizing Permuted Activations for Interpolation Repair, September 2023. URL http://arxiv.org/abs/2211.08403. arXiv:2211.08403 [cs.LG].

Jonathan Kahana, Eliahu Horwitz, Imri Shuval, and Yedid Hoshen. Deep Linear Probe Generators for Weight Space Learning, October 2025. URL http://arxiv.org/abs/2410.10811. arXiv:2410.10811 [cs.LG].

Miltiadis Kofinas, Boris Knyazev, Yan Zhang, Yunlu Chen, Gertjan J. Burghouts, Efstratios Gavves, Cees G. M. Snoek, and David W. Zhang. Graph Neural Networks for Learning Equivariant Representations of Neural Networks, July 2024. URL http://arxiv.org/abs/2403.12143. arXiv:2403.12143 [cs.LG].

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of Neural Network Representations Revisited, July 2019. URL http://arxiv.org/abs/1905.00414. arXiv:1905.00414 [cs.LG].

Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence, June 2015. URL http://arxiv.org/abs/1411.5908. arXiv:1411.5908 [cs.CV].

Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent Learning: Do different neural networks learn the same representations?, February 2016. URL http://arxiv.org/abs/1511.07543. arXiv:1511.07543 [cs.LG].

Derek Lim, Haggai Maron, Marc T. Law, Jonathan Lorraine, and James Lucas. Graph Metanetworks for Processing Diverse Neural Architectures, December 2023. URL http://arxiv.org/abs/2312.04501. arXiv:2312.04501 [cs.LG].

Derek Lim, Theo Moe Putterman, Robin Walters, Haggai Maron, and Stefanie Jegelka. The Empirical Impact of Neural Parameter Symmetries, or Lack Thereof, October 2024. URL http://arxiv.org/abs/2405.20231. arXiv:2405.20231 [cs.LG].

Shiwei Liu, Tianlong Chen, Xiaohan Chen, Li Shen, Decebal Constantin Mocanu, Zhangyang Wang, and Mykola Pechenizkiy. The Unreasonable Effectiveness of Random Pruning: Return of the Most Naive Baseline for Sparse Training, February 2022. URL http://arxiv.org/abs/2202.02643. arXiv:2202.02643 [cs.LG].

Eran Malach, Gilad Yehudai, Shai Shalev-Shwartz, and Ohad Shamir. Proving the Lottery Ticket Hypothesis: Pruning is All You Need, February 2020. URL http://arxiv.org/abs/2002.00585. arXiv:2002.00585 [cs.LG].

Aviv Navon, Aviv Shamsian, Idan Achituve, Ethan Fetaya, Gal Chechik, and Haggai Maron. Equivariant Architectures for Learning in Deep Weight Spaces, January 2023. URL https://arxiv.org/abs/2301.12780v2.

Aviv Navon, Aviv Shamsian, Ethan Fetaya, Gal Chechik, Nadav Dym, and Haggai Maron. Equivariant Deep Weight Space Alignment, November 2024. URL http://arxiv.org/abs/2310.13397. arXiv:2310.13397 [cs.LG].

Ameya Prabhu, Girish Varma, and Anoop Namboodiri. Deep Expander Networks: Efficient Deep Networks from Graph Theory, July 2018. URL http://arxiv.org/abs/1711.08757. arXiv:1711.08757 [cs.CV].

Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's Hidden in a Randomly Weighted Neural Network?, March 2020. URL http://arxiv.org/abs/1911.13299. arXiv:1911.13299 [cs.CV].

Konstantin Schürholt, Boris Knyazev, Xavier Giró-i Nieto, and Damian Borth. Hyper-Representations as Generative Models: Sampling Unseen Neural Network Weights, September 2022a. URL http://arxiv.org/abs/2209.14733. arXiv:2209.14733 [cs.LG].

Konstantin Schürholt, Diyar Taskiran, Boris Knyazev, Xavier Giró-i Nieto, and Damian Borth. Model Zoos: A Dataset of Diverse Populations of Neural Network Models, September 2022b. URL http://arxiv.org/abs/2209.14764. arXiv:2209.14764 [cs.LG].

Konstantin Schürholt, Michael W. Mahoney, and Damian Borth. Towards Scalable and Versatile Weight Space Learning, June 2024. URL http://arxiv.org/abs/2406.09997. arXiv:2406.09997 [cs.LG].

Sidak Pal Singh and Martin Jaggi. Model Fusion via Optimal Transport, May 2023. URL http://arxiv.org/abs/1910.05653. arXiv:1910.05653 [cs.LG].

N. Joseph Tatro, Pin-Yu Chen, Payel Das, Igor Melnyk, Prasanna Sattigeri, and Rongjie Lai. Optimizing Mode Connectivity via Neuron Alignment, November 2020. URL http://arxiv.org/abs/2009.02439. arXiv:2009.02439 [cs.LG].

Thomas Unterthiner, Daniel Keysers, Sylvain Gelly, Olivier Bousquet, and Ilya Tolstikhin. Predicting Neural Network Accuracy from Weights, April 2021. URL http://arxiv.org/abs/2002.11448. arXiv:2002.11448 [stat.ML].

Elena Voita and Ivan Titov. Information-Theoretic Probing with Minimum Description Length, March 2020. URL http://arxiv.org/abs/2003.12298. arXiv:2003.12298 [cs.CL].

Yilun Xu, Shengjia Zhao, Jiaming Song, Russell Stewart, and Stefano Ermon. A Theory of Usable Information Under Computational Constraints, February 2020. URL http://arxiv.org/abs/2002.10689. arXiv:2002.10689 [cs.LG].

Greg Yang and Edward J. Hu. Feature Learning in Infinite-Width Neural Networks, July 2022. URL http://arxiv.org/abs/2011.14522. arXiv:2011.14522 [cs.LG].

Boya Zeng, Yida Yin, Zhiqiu Xu, and Zhuang Liu. Generative Modeling of Weights: Generalization or Memorization?, October 2025. URL http://arxiv.org/abs/2506.07998. arXiv:2506.07998 [cs.LG].

Allan Zhou, Kaien Yang, Kaylee Burns, Adriano Cardace, Yiding Jiang, Samuel Sokota, J. Zico Kolter, and Chelsea Finn. Permutation Equivariant Neural Functionals, September 2023. URL http://arxiv.org/abs/2302.14040. arXiv:2302.14040 [cs.LG].

Allan Zhou, Chelsea Finn, and James Harrison. Universal Neural Functionals, February 2024. URL http://arxiv.org/abs/2402.05232. arXiv:2402.05232 [cs.LG].

A ZOOS: ARCHITECTURES, TRAINING AND STORAGE

Architecture. Every network is a three layer perceptron on flattened CIFAR-10 images scaled to $[-1, 1]$: an input layer of 3072 units, two hidden layers of equal width with GELU activations and biases, and a linear head onto the two groups of the labelling. Widths are 32, 64, 128, 256 and 512. Throughout the paper, "first layer" is the input to hidden map and "second layer" the hidden to hidden map; biases and the head are trained but never stored or read. Weights are initialised with the PyTorch default for linear layers, uniform on $\pm 1/\sqrt{\text{fan-in}}$ with the fan in of the dense layer, in both families; masked entries are then set to zero, so a masked unit starts with smaller pre-activations than a dense one. Appendix F.4 reports how far each layer travels from this initialisation.

Tasks. A task is a surjective map from the ten CIFAR-10 classes to two groups. We draw 20 distinct such maps, none of which is another with its group labels swapped. Group sizes range from five and five to eight and two, so the majority baseline differs between tasks, and accuracy is best read against it. Each task is trained from the same 50 initialisations: seed s fixes both the initial weights and the order of the data, so the design is paired across tasks. This gives 1,000 networks per family and width.

Training. All networks see the same fixed subset of $8{,}000$ training images and the full test split, for 20 epochs of AdamW at learning rate 10^{-3}, weight decay 10^{-4} and batch size 256. Nothing but the labelling and the seed differs between networks of a family. Test accuracies range from about 0.59 to about 0.79 across tasks and are stable across widths; the tasks differ in difficulty, and that difference is the footprint that Section 3 removes before measuring structure.

Families. In the masked family the two hidden layers carry a fixed sparse mask with row regular connectivity: every output unit receives exactly $d = \max(3, \mathrm{round}(0.1 \times \text{fan-in}))$ inputs, that is 307 in the first layer and 3, 6, 13, 26 and 51 in the second. The mask is drawn once per architecture from a fixed seed and reused for every task, every seed and every epoch, so a given unit reads the same inputs in every network of that width; because rows are drawn in order from one generator, the first layer mask of a narrow network is the first rows of the mask of a wider one. Row regularity matters at the narrow end: sampled globally at the same density, several units of a width 32 layer would receive no inputs at all and be dead by construction. The masks are row regular, not spectrally certified, and we make no claim about their spectra. The dense family is the same protocol with the mask set to all ones.

Null arm. For the masked family at every width, and the dense family at width 128, we train a matching collection in which all 20 groups share a single labelling and are formed from disjoint blocks of seeds, so group j uses seeds $50j + 1$ to $50(j + 1)$. The task is constant by construction, so every quantity we report has a floor measured on real data with the same code. An earlier version of this arm trained all groups from the same seeds, which makes them bitwise identical and turns the floor into an arithmetic zero rather than an estimate; the disjoint blocks are what make it a measurement.

Storage. For each network we store the initial and final values of a fixed uniform random subset of each layer's live coordinates, capped at $20{,}000$ per layer and identical across all networks of a width, together with the test accuracy. The cap is a storage decision: a uniform subsample of a uniform subsample is still uniform, so a probe on n stored coordinates is a probe on n coordinates of the layer, but the denominator has to be reconstructed from the layout rather than taken from the stored vector, which is what Appendix B does. For the dense family at every width, and for the masked family at width 128, we additionally store full weights, the first layer in half precision, which is what the invariant readers, the aggregate reader and the matching require. The images used for matching are $2{,}560$ images of the test split, identical for every network, with labels never used: the first ones compute the matching and the last 512 score its quality.

Splits. Every probe splits networks by initialisation seed, with 60% of seeds for training, 20% for validation and the rest for test. No seed appears in two splits. Where a matching template is fitted, it is fitted on the training seeds only and then frozen, using the same seed permutation as the probe, so no test network takes part in building the frame in which it is later read. Because a template belongs to one split, the matched frame is measured on that split only; every other budget is repeated over ten splits.

B THE READING BUDGET: ESTIMATOR AND VALIDATION

Definition. For a given reader the probe is run over a geometric ladder of budgets $n \in \{1, 2, 4, \ldots\}$ up to the number stored. The budget n^* of Equation 1 is the smallest n at which the probe clears the bar, interpolated linearly in $\log n$ between the two budgets that bracket the crossing. When the smallest budget tried already clears the bar the value is reported as an upper bound; when the curve never crosses, the value is infinite, which is informative and is never silently replaced by the largest budget tried.

The bar. The bar is 0.95 for every reader and every width. We arrived at it from a relative bar, 0.95 times the ceiling of the weakest width, which asks the same question at every width while never setting a target some width cannot reach. On the first layer every reader we report has a ceiling of 1.0 at every width, so the two coincide, and the absolute form lets readers be compared on one axis. The one exception, the coordinate probe on dense layers read directly, peaks at 0.40 and is reported

as infinite. Under a relative bar its own low ceiling would set its target near 0.4 and make it look cheap, which is the reason the bar is absolute. Two alternatives were rejected. A bar pegged to each width's own ceiling asks a different question at each width whenever the ceilings differ, so we report it only as a sensitivity. A bar pegged to the whole vector probe is worse still: with several hundred training networks against tens of thousands of coordinates, the whole vector probe degrades as the layer grows, so pegging to it would let a failing reference masquerade as an easy target. For the second layer, whose ceilings differ from 0.32 to 1.0 across widths, the relative and own bars disagree, and we make no budget claim there.

The aggregate reader. For each network, per pixel means are taken over the first u units of a fixed random order of units, for u on a geometric ladder up to the width, and the probe reads m pixels of a fixed random order, for m on a geometric ladder up to 3072. Cell (m, u) costs mu coordinates, and n^* is the cheapest cell that clears the bar. The unit order, the pixel order and the split all change with the seed. On synthetic networks with the task planted in the unit average and rows permuted per seed, it finds the planted signal; on a planted null, no cell clears the bar.

The axis. Budgets are fitted against the number of live weights the layer actually has, $N = \text{width} \times d$ with d as in Appendix A, not against the number of coordinates stored. Because storage is capped, the stored count saturates and three of five widths share the same value, so a fit against it regresses on tied points and returns nonsense.

The verdict. We fit $\log n^*$ against $\log N$ by least squares and report the slope with a 95% interval from the residual t quantile, and, separately, the spread of per split slopes over ten seed splits. The two answer different questions: the split spread is the instrument's noise, the t interval is the misfit of a power law to five widths. Under fixed wiring the split spread is narrow and the t interval wide, which means the shape across widths repeats from split to split and is not a power law; this is why we report per width medians with their range and describe the budget as showing no net growth rather than as constant. We never read the verdict from R^2: a flat noisy line has $R^2 \approx 0$ by construction, so gating on it refuses to conclude exactly when the answer is flatness.

Self test, and the design errors it caught. The estimator is run on synthetic data with a planted ground truth in two regimes: a fixed number of informative coordinates at every width, where the truth is a constant budget, and a fixed fraction, where the truth is proportional. It separates them, and in doing so it caught two errors before any real measurement. First, a bar defined as a fraction of the whole vector probe's accuracy is a falling target, because that accuracy degrades with layer size at fixed model count; it made a planted proportional budget look constant. Second, gating the verdict on R^2 refused to return the constant verdict on data where the constant was planted. The invariant readers have their own self test, on synthetic layers whose rows are permuted per seed: they recover the planted task, and a linear coordinate probe recovers part of it too, because a linear probe over all coordinates can form sums over units; this is why the coordinate row of Table 1 sits above chance.

A statistic we do not rely on. The effective number of informative coordinates, computed as a participation ratio over per coordinate scores, returns an exponent near 0.8 on planted data where the truth is zero, because most of the scores it averages are noise floor. We do not report it.

C CONTROLS, IN FULL

Each control below changed a number we had already believed. They are reported in the order in which they bit.

Per model scale. Tasks differ in difficulty, difficulty shapes the loss trajectory, and the trajectory sets how far weights travel, so summary statistics of a layer carry task identity on their own (Table 4). On the public zoo of Section 9 four per layer norms identify the training dataset almost perfectly, which is why we treat that benchmark as saturated. Every decode result in this paper is measured after standardising each network's layer over its live weights. Where the comparison is informative we report both: for the dense first layer the raw and standardised probes agree to within noise, so nothing there is scale.

Table 4: Task identity recovered from per layer summary statistics alone, on the relabelled zoo, masked family at width 256. Chance is 0.05. A handful of scalars recovers a large part of the task, which is why every result in this paper is measured after removing per model scale.

Features	Dimension	Probe
Log norm, first layer	1	0.200
Log norm, second layer	1	0.227
Both norms	2	0.450
Both means	2	0.413
Both standard deviations	2	0.427
Norms, means and deviations	6	0.737

Structural zeros. In a masked layer most entries are zero by construction. Standardising the layer with those entries included maps each of them to minus the mean over the standard deviation, a per model scalar copied into thousands of fixed positions, which reintroduces exactly the footprint the standardisation was meant to remove. Doing this inflated the unaligned second layer probe to roughly three times chance and a permutation invariant summary of it to eight times chance. Standardising live entries only returns both to their correct values, near chance and around seven times chance respectively.

Information carried by the matching. A matching is computed from a network's own activations, so it is a function of that network's task, and weights read in a matched frame can carry the task even when the weights in that frame carry nothing task specific. We test this four ways. (i) The matchings themselves, encoded as permutation matrices, are given to the probe. In the masked family the layer one matching alone identifies the task at 0.875, because a masked unit is pinned to fixed inputs and so its index is comparable across networks; in the dense family the same quantity is at chance. (ii) Each network's own matching is applied to a single fixed matrix that carries no task. This stays at chance everywhere, including in the masked family where we know from (i) that the permutation carries the task, so the control is a lower bound on the leak and cannot clear a layer on its own; we report it as uninformative rather than as evidence. (iii) Each half of the alignment is applied alone. In the dense family neither half works without the other. (iv) The swap control, below.

The swap control, and why it is invalid on real models. Two networks of the same family that share an initialisation should, if training moves units little, share a frame. That would let us exchange the two ingredients of a matched reading: one network's weights in another's frame isolates the values, and the converse isolates the matching. We measured whether the assumption holds, by scoring a partner's matching on a network's own held out activations against that network's own matching. It does not: the partner's matching fits barely above a random permutation. The swap is therefore uninformative on these models, and we say so rather than reading its numbers. This is the control that would have settled the attribution question in the deeper layer, and it is why that question is left open in Section 8.

The pipeline itself. Every control above is run on the null arm, where all networks share a task. There, the same code with the same template and the same splits stays between chance and twice chance on every arm, including the aligned ones, while the corresponding relabel numbers run from a third to all of the way to one. The same holds for the readers that need no correspondence: on the dense null arm at width 128 every invariant reader, the weight matching and every cell of the aggregate reader stay at or below 0.10, and on the masked null arm the coordinate probe never exceeds 0.09 at any width. Notably, matching quality is slightly higher on the null arm than on the relabel arm, which is what task dependent matching predicts: with the task held fixed, units correspond a little better across networks.

Self tests. The matching pipeline is validated on synthetic networks in three regimes: task structure planted in the second layer, no task structure at all, and task structure planted in the first layer only. It recovers the first, returns chance in the second, and in the third attributes the aligned second

layer's apparent signal to the matching rather than to the weights. Two design errors were caught there before any real run: a partner drawn as "task plus one", which makes the partner's task a deterministic function of the target and lets the control decode through it, and swap controls computed only at the largest matching batch, which left the small batch regime, where the leak actually lives, uncontrolled.

Features that were not invariant. The permutation invariant features of Section 9 summarise each layer over its hidden axes and keep its fixed axes: input channels and kernel offsets in the first layer, classes in the last. Dropping the classifier head, so that datasets with different class counts share one architecture, made the last kept layer look like an output layer, and its hidden units were treated as a fixed axis. The features then moved by 2.1 under a permutation that leaves the function unchanged. Marking the last kept layer as hidden restores exact invariance. The numbers in Table 3 are from the corrected features, with the regularisation strength chosen on validation runs rather than on the test split. The two corrections together raised the invariant features from 0.942 to 1.000 and lowered the norms from 0.983 to 0.973: the benchmark is still saturated by four scalars, but our earlier statement that the scalars beat the full features did not survive.

A ranker that was not what it said. The label free "movement" ranker orders coordinates by how far they travelled from initialisation. In the matched frame the initial weights are not aligned, so the aligned zoo is written with a zero initialisation and the ranker reduces to the magnitude of the final weight. Table 10 labels it accordingly.

D THE ADAPTATION EXPERIMENT

Setup. A masked source network of width 256 is pretrained on the ten way CIFAR-10 task, then fine tuned on each of four relabellings held out from everything used to rank coordinates, ten seeds each, with every arm paired by target and seed. Readability is the between task share of variance of each first layer coordinate, estimated on fifteen other relabellings trained from scratch; the source relabelling and the targets are disjoint from them. Arms restrict which parameters may move: the most readable 12% of first layer coordinates, an equal sized random subset, the least readable 12%, the whole network, and the head alone. In every restricted arm and in the head only arm the head is trainable and every other weight and bias is frozen; frozen values are restored from the pretrained tensors after every optimiser step and asserted unmoved. A from scratch arm trains at the same budget.

The stopping rule, and the deviation. We registered that the experiment would be void unless head only fine tuning trailed full fine tuning by at least three points of accuracy. It trails by 1.5, so the rule fired: in accuracy, the head alone does most of what the body does, and a comparison of body subsets on accuracy has little room. We therefore report the comparison on prequential code length, which resolves differences accuracy cannot, as a declared deviation from the registered analysis. An earlier version of the experiment left biases trainable in every arm, so its "head only" arm also moved every body bias; freezing them cost the head only arm about three bits and changed no conclusion, and the full and from scratch arms, which the change does not touch, reproduce the earlier run exactly.

Why the source task is ten way. With a relabelling as the source, from scratch training beat every fine tuning arm on all four targets, and the arms were being ranked by how quickly they escaped a harmful initialisation. Two binary partitions of the same ten classes agree on roughly half the classes and disagree on the rest, and a binary head can only flip globally, so resolving a partial disagreement requires the features themselves to move. The ten way task refines every binary relabelling of it, so transfer is positive. The same choice also limits the experiment: when the source features already suffice, the first layer needs to change little, which is part of why the head alone does so well.

Why the pool is one layer. With the whole body as the selection pool, the most readable coordinates were entirely first layer while the least readable were mostly second layer, because almost all second layer coordinates sit at the floor of the ranking. The apparent ordering was a layer effect, and it reversed when the pool was restricted to the first layer.

Instrument and resolution. We report prequential code length, the cumulative log loss of the online predictor over the first pass, in bits, paired by target and seed against the head only arm (Table 5). The restricted first layer is worth about 29 bits over the head alone, whichever coordinates it contains; the most readable and the random subset differ by 0.3 bits, against standard errors of 1.3 bits on each arm's paired contrast. The selection effect is therefore below a tenth of what the restricted layer contributes, and full fine tuning, at 209 bits, shows the instrument has range to spare.

Table 5: Fine tuning restricted to a fixed fraction of the first layer, forty paired runs per arm. Code length is prequential over the first pass, in bits; the last column is paired against the head only arm, with its standard error. Lower is better. Accuracies are medians.

Arm	Accuracy	Code length	Against head only
Most readable 12%	0.6836	7421	-28.7 ± 1.3
Random 12%	0.6810	7421	-29.0 ± 1.3
Least readable 12%	0.6807	7420	-29.4 ± 1.4
Full fine tuning	0.6900	7218	-209.4 ± 24.0
Head only	0.6745	7441	0
From scratch	0.6778	7705	$+204.1 \pm 25.6$

E Registered predictions and their outcomes

Every experiment in this paper was preceded by a written prediction with decision rules, registered before the run. We list them with what happened, including the ones that failed, because the failures are what shaped the paper. The lower block lists the predictions registered for the experiments added in the final revision.

Two patterns in the failures are worth stating. Most of them underestimated how much of the task is recoverable, but not all: we overestimated how much a width dependent initialisation scale would change what training moves, and we expected the linear half of the invariant reader to fail on dense layers, which it did not. And each failure was a failure of attribution rather than of detection. The information was there; what we had wrong was where it came from, and each correction came from a control or a new reader rather than from more of the same measurement.

F Additional results

F.1 Task identification by family and width

Figure 5 states the comparison of Figure 2 in a single view: the task is identified perfectly from the first layer under fixed wiring, poorly by a coordinate probe on the same layer trained dense, and perfectly again once the dense networks' hidden units are matched. Table 1 shows that the poor middle bar is a property of the coordinate reader, not of the layer.

F.2 Budgets by reader

Table 7 gives the numbers behind Figure 3, at the bar of Equation 1. Ranges are over seed splits: ten for fixed wiring and three for the aggregate reader; the matched frame is measured on the split its template was built on.

F.3 The aggregate reader, cell by cell

Table 8 shows part of the aggregate reader's grid for the dense first layer at width 128. Accuracy rises with the number of pixels read and, much more steeply, with the number of units averaged; the cheapest cell that clears the bar reads 64 pixels over all 128 units, and no cell with fewer units than the width does so more cheaply. At a fixed number of units, accuracy falls as the layer widens: with 64 pixels over 64 units it is 0.88 at width 128, 0.74 at width 256 and 0.79 at width 512, so each unit carries a smaller share of the average in a wider layer.

Table 6: Registered predictions and outcomes. "Held" means the registered branch fired as written.

Prediction	**Outcome**
The reading budget falls with layer size	Failed: no net growth, and not monotone
Label free rankings track the supervised one	Failed: they track random under fixed wiring
Per layer summary statistics stay near chance on the relabelled zoo	Failed: six scalars reach fifteen times chance
The layer mean alone stays near chance	Failed: eight times chance
After standardisation the first layer still decodes perfectly	Held
After standardisation the random subset budget grows less than twofold	Held between the end widths; the curve is not monotone
A permutation invariant summary of the second layer is weak	Failed: stronger than predicted, and grows with width
Matched second layer reading lands in a middling range	Failed: higher, and rises with width
The swap control is valid on same initialisation pairs	Failed: partner matchings fit barely above random
Dense first layer decodes as the masked one does	Failed for the coordinate probe; see the invariant readers below
Dense budgets land within twofold of masked ones	Failed at four of five widths
Matching restores the dense first layer	Held
The budget in the matched dense frame is flat	Held for ranked coordinates; random subsets grow weakly
Layer one matchings carry nothing in the dense family	Held
Every arm of the null pipeline stays at its floor	Held, in both families
Head only fine tuning trails full fine tuning by three points (stopping rule)	Failed: 1.5 points; the rule fired
The most readable subset beats random by two points	Failed: within 0.3 points
Distance from initialisation is flat in width in the first layer and grows as the square root of width in the second	Failed: it falls in the first and grows far less in the second
Invariant readers decode the dense first layer at 0.9 or above at every width	Held: 1.000 at every width
Singular values of the dense first layer decode between 0.15 and 0.40	Failed at widths 256 and 512: 0.48
Weight matching with no images restores the dense first layer	Held
Per pixel means alone stay at or below 0.6 on the dense first layer	Failed: 1.000
Over ten splits the masked budget exponent stays within ± 0.3, and its split band does not exclude 0.5	Held for the first part; failed for the second, the band excludes 0.5
Without correspondence the cheapest reading uses every unit and grows with width, exponent between 0.7 and 1.2	Held: every unit, exponent 0.70 to 0.80
No cell of the aggregate reader clears the bar on the null arm	Held

F.4 Distance travelled from initialisation

Table 9 reports, for each layer, family and width, the median over networks of $\|W_{\text{final}} - W_{\text{init}}\| / \|W_{\text{init}}\|$ on the stored coordinates. Three facts matter for the main text. In the first layer every network travels several times its initial norm, so its weights are set by training rather than by initialisation in both families; the distance falls with width, and masked layers travel 1.6 to 1.9 times further than dense ones. In the second layer of dense networks the distance grows from 0.60 to 1.06 of the initial norm, so the share of the layer set by training roughly doubles across our widths, which coincides with the width trend of Section 8. And the null arm travels as far as the relabelled one, so distance is a property of training, not of how different the tasks are. The width dependence follows the standard picture for this parameterisation and optimiser (Yang & Hu, 2022; Everett et al., 2024), although the second layer grows much less than a fixed step against a shrinking initialisation would predict.

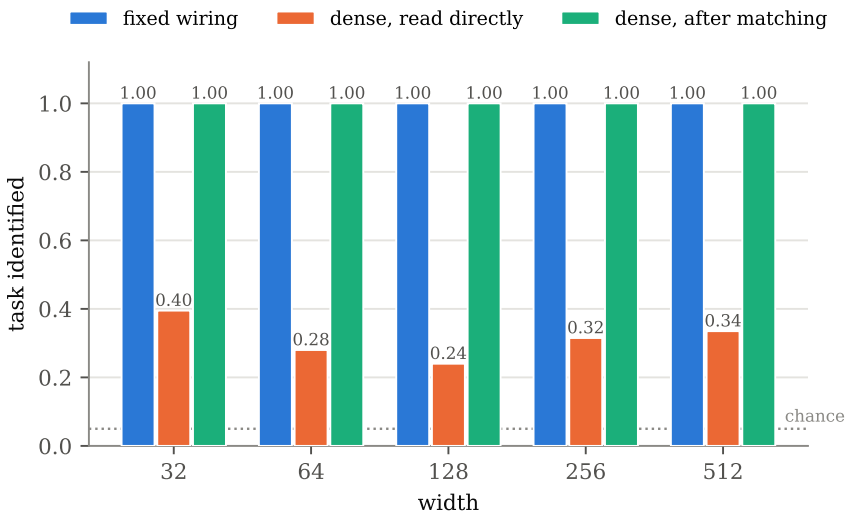


Figure 5: First layer task identification by a coordinate probe for the masked family, the dense family read directly, and the dense family after matching, at each width. Chance is the dotted line.

Table 7: First layer budget n^* at accuracy 0.95. Medians, with the range over seed splits in brackets where there is one. N is the number of live weights. The premium is the aggregate reader's budget over the matched frame's random budget.

Width	32	64	128	256	512
N, masked	9,824	19,648	39,296	78,592	157,184
Fixed wiring, ranked	219	87	80	103	200
Fixed wiring, random	373 [229, 676]	164 [138, 210]	168 [121, 185]	226 [210, 279]	419 [342, 512]
N, dense	98,304	196,608	393,216	786,432	1,572,864
Dense, read directly	∞	∞	∞	∞	∞
Dense, aggregate reader	2,048	4,096	4,096 [4k, 8k]	8,192 [8k, 16k]	16,384
Dense, matched, ranked	431	304	456	274	161
Dense, matched, random	619	676	512	765	967
Premium	3.3	6.1	8.0	10.7	16.9

F.5 Rankers

Table 10 compares coordinate rankings at a small budget. The supervised rankings beat a random subset of the same size at every width in both families. The label free ones, which are the only rankings available to an encoder that does not yet know the task, track random in the masked family and beat it in the matched dense frame, where the column labelled magnitude is the ranker we had called movement (Appendix C).

Table 8: Aggregate reader on the dense first layer at width 128, first seed split: test accuracy for m pixels (rows) averaged over u units (columns). The reader costs mu coordinates; the bar is 0.95.

$m \backslash u$	1	8	32	64	128
16	0.065	0.195	0.370	0.525	0.775
32	0.095	0.175	0.490	0.685	0.940
64	0.110	0.225	0.670	0.880	0.990
128	0.110	0.310	0.725	0.930	1.000
3072	0.120	0.400	0.845	1.000	1.000

Table 9: Median distance from initialisation relative to the initial norm, stored coordinates, one thousand networks per cell. The dense null arm exists at width 128 only.

Layer	**Arm**	32	64	128	256	512
First, masked	relabel	6.74	6.17	5.49	4.93	4.25
First, masked	null	7.15	6.48	5.74	5.11	4.39
First, dense	relabel	3.59	3.47	3.21	2.93	2.66
First, dense	null			3.13		
Second, masked	relabel	1.98	2.22	2.37	2.47	2.43
Second, masked	null	2.14	2.35	2.50	2.55	2.48
Second, dense	relabel	0.60	0.70	0.79	0.92	1.06
Second, dense	null			0.77		

Table 10: Task identification from the thirty two highest ranked coordinates of the first layer, chance 0.05. Upper block: masked family. Lower block: dense family in a matched frame, where the last ranker reads the magnitude of the final weight because the aligned zoo carries no aligned initialisation.

Width	**ICC**	**between**	**variance**	**mean abs.**	**movement / magnitude**	**random**
32	0.770	0.635	0.405	0.470	0.405	0.500
64	0.835	0.775	0.490	0.585	0.500	0.435
128	0.775	0.760	0.450	0.470	0.510	0.610
256	0.715	0.695	0.595	0.500	0.570	0.480
512	0.580	0.530	0.460	0.330	0.485	0.340
32	0.605	0.600	0.430	0.415	0.445	0.260
64	0.570	0.550	0.335	0.295	0.345	0.245
128	0.590	0.535	0.360	0.330	0.395	0.275
256	0.600	0.630	0.390	0.285	0.310	0.235
512	0.695	0.690	0.475	0.315	0.440	0.200