TARGET SCALE IS READOUT SCALE: ONE RATIO ORDERS HOW LONG COLLAPSE LASTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Many self-supervised methods train an encoder through a linear readout that regresses a dense target. When the encoder ends in a layer normalization, the readout's output at initialization has a norm set by the widths alone, regardless of the target. Scaling the target is therefore equivalent to scaling the readout, and what matters is the ratio r between the readout's initial output and the target's norm. The standard recipe, a unit-gain layer normalization with a fan-in initialized readout and a unit-variance target, fixes $r \approx 1/\sqrt{3}$ at every width. We test what happens when it does not, in a controlled case study: MelHuBERT trained on LibriSpeech with regression targets of different scales. Rank collapsed early in every run with a default readout, and r ordered how long the collapse lasted: about a thousand steps at $r = 0.14$ and more than twelve thousand at $r = 37$. Of 94 pairs of runs at different ratios, 90 are ordered as r predicts, none against it, and 4 are undetermined because neither run had recovered. Most of the collapse is a shared offset, with 98% of the representation's energy in its mean over frames; with the mean removed rank falls by over a third, and further under a softmax readout, which builds no offset. At 8,500 steps r orders both the offset and the rank that remains without it. In 500-step runs at $r = 37$, a sixty-fold change in the readout's learning rate moved rank by two, while lowering r restored 100 to 419 dimensions. At large r the gradient at the encoder's output is dominated by a term that does not contain the target, and an approximate closed form for its direction matches simulation to 0.3%. MAE (0.89) and data2vec (0.58 to 1.63) sit close to $1/\sqrt{3}$. Distance from initialization measured by CKA, a common diagnostic, can miss the collapse: CKA centers the representation, and two runs 0.12% apart on it differ sevenfold in rank, twofold with the mean removed. Checking r costs one batch, before training starts.

1 INTRODUCTION

Many representation-learning systems train their encoder through a linear readout, a single linear layer that maps the encoder's output to the prediction and is discarded after training. Masked autoencoders regress normalized pixels through one (He et al., 2022), data2vec regresses a teacher's averaged layers through one (Baevski et al., 2022), and distillation and feature-prediction objectives do the same with other targets. In most of these systems the encoder's last operation is a layer normalization (Ba et al., 2016; Xiong et al., 2020).

A unit-gain layer normalization hands the readout a vector of norm $\sqrt{d}$ for every input. A readout drawn at the usual fan-in scale then produces, at initialization, an output whose norm depends only on the widths: about $\sqrt{m/3}$ for an m-dimensional target under PyTorch's default. The target's scale is therefore not set by the task alone. It is the denominator of a ratio,

$$r \;=\; \frac{\mathbb{E}\,\|W_0 h + b_0\|}{\mathbb{E}\,\|y\|},$$

and a small target is a large r. Equivalently, scaling a target down is the same as scaling its readout up (Section 2).

The standard recipe hides this. A unit-gain layer normalization, a fan-in initialized readout and a target normalized to unit variance per coordinate give $r \approx 1/\sqrt{3}$ at every width, and the published

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.

systems we checked sit between 0.58 and 1.63. To our knowledge this identity has not been stated, though it follows directly from standard initialization formulas. To see what r does, we moved it deliberately, by changing the target's scale in a speech encoder trained by masked prediction (Section 3).

With a default readout, the encoder's rank collapsed early in every run: in the two runs we recorded step by step, from 518 to 26 dimensions within twenty steps (Figure 1a). Most of that is a shared offset: by step 50 the mean over frames carries 98% of the representation's energy (Section 4.2); with the mean removed rank falls by over a third, and further under a softmax readout, which builds no offset. What r changed was not whether the collapse happened but how long it lasted, from about a thousand steps at $r = 0.14$ to more than twelve thousand at $r = 37$, with no pair of runs recovering against the order of their ratios (Figure 1b, Section 4.1). The two target shapes we tested at matched r recovered together, and rescaling a single target moved rank as its new r predicts. In 500-step runs the readout's learning rate mattered less: a sixty-fold change at $r = 37$ moved rank by two, where lowering r moved it by 100 to 419 (Section 4.3).

At large r the gradient at the encoder's output is dominated by $-W^\top W h$ (Section 4.4). The target is not in that term, and neither is the learning rate once Adam normalizes the step; an approximate closed form gives its direction from the two widths. A readout that starts too large barely moves relative to its own size, and the loss passes through the loss of a constant predictor while the representation is dominated by one shared vector. How hard the readout's scale pushes depends on the loss: quadratically under squared error, at most linearly under a softmax, and not at all under a cosine.

Our contributions are:

- **The ratio.** Behind a unit-gain layer normalization, scaling the target is equivalent to scaling the readout, and the standard recipe fixes the resulting ratio near $1/\sqrt{3}$ at any width (Section 2).
- **Its effect, measured.** In one model and corpus, r orders how long rank collapse lasts. Most of the collapse is a shared offset, and r orders both the offset and the rank that remains without it; at matched r target shape did not change the order, and the readout's learning rate mattered less than r (Sections 4.1 to 4.3).
- **The mechanism at initialization.** At large r, the gradient at the encoder's output is dominated by a term that does not depend on the target, with an approximate closed-form direction, and the loss sets how strongly readout scale pushes (Section 4.4).
- **A blind spot.** Distance from initialization, a common way to ask how much training changed a representation, can miss the collapse because it centers the representation: two runs 0.12% apart on it differ sevenfold in rank, and twofold with the mean removed (Section 4.5).

The algebra is general; the training evidence comes from one model family, so we establish a controlled failure mode, not how often it occurs. Before several rounds of runs we wrote down what we expected; five of the nine predictions were refuted (Appendix F). Code and every measurement are in the supplementary material, with one command that rebuilds every number, table and figure.

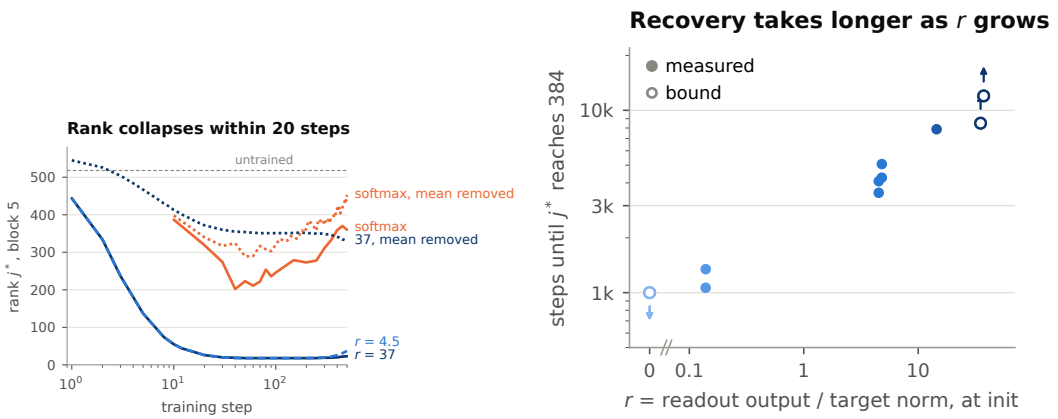
2 The readout-to-target ratio

2.1 The readout's first output does not depend on the target

Let $h \in \mathbb{R}^d$ be the encoder's output at one input position (a frame, in our case study) and let the readout be $f(h) = Wh + b$ with $W \in \mathbb{R}^{m \times d}$, trained on $\frac{1}{2}\|f(h) - y\|^2$. At initialization a layer normalization has unit gain and zero bias, so it leaves each position with mean zero and unit variance across its coordinates: $\|h\|^2 = d$, up to the normalization's small ϵ. PyTorch's default draws every entry of W and b from $\mathcal{U}(-1/\sqrt{d}, 1/\sqrt{d})$, with variance $1/(3d)$. Then, for every input,

$$\mathbb{E}\,\|W_0 h + b_0\|^2 \;=\; \frac{m}{3}\Big(1 + \frac{1}{d}\Big), \tag{1}$$

with the expectation over the draw of W_0, b_0 (Appendix A.1). The norm of a sum of many independent terms concentrates, so $\mathbb{E}\|W_0 h + b_0\|$ is close to the square root of Equation 1: about $\sqrt{m/3}$,



(a) (b)

Figure 1: **(a)** Rank (j^*, Section 3) at block 5 over the first 500 steps, for two regression targets eight times apart in r and for a softmax readout, for which r is undefined. The two regression runs fall from 518 to 26 within twenty steps and agree within 1 in j^* for 200 steps. Dotted: rank with the mean over frames removed, for $r = 37$ and the softmax. Step axis logarithmic. **(b)** Steps until rank recovers to half its ceiling, against r, both axes logarithmic. Filled points are measured; open points are bounds (recovered before the first reading, or not recovered by the last).

which is 5.164 for $m = 80$. Our own readout at seed 0 reads 5.281 over 120,000 real frames; the gap comes from a single draw of W_0 meeting inputs that are not isotropic. The target appears nowhere. We define

$$r = \frac{\mathbb{E}\,\|W_0 h + b_0\|}{\mathbb{E}\,\|y\|}. \tag{2}$$

r is a squared-error quantity; a softmax readout has no target norm.

2.2 Scaling the target is scaling the readout

Training on $s\,y$ with readout (W, b) and training on y with readout $(W/s, b/s)$ give the same residual up to a factor s, so the encoder's gradient differs only by s^2 and the readout's by s (Appendix A.2). Adam (Kingma & Ba, 2015) divides out a constant gradient scale whenever its ϵ is small next to the gradients, so the encoder's update is unchanged, and the readout's update matches once its learning rate is also divided by s. Scaling a target thus changes two things about the readout at once: its initial size and its effective learning rate. Section 4.3 separates them. This is the symmetry between initialization, multipliers and learning rates of Everett et al. (2024), applied to the target instead of a layer. It holds up to Adam's ϵ: a run with $\epsilon = 10^{-14}$ matches the default run's rank at 8,500 steps (98 and 54 against 106 and 55 on the two seeds).

2.3 The standard recipe fixes the ratio

Normalize the target to unit variance per coordinate, so that $\mathbb{E}\|y\|^2 = m$. If its norm also concentrates near $\sqrt{m}$, as it does for targets normalized per example, then with Equation 1

$$r \;\approx\; \frac{\sqrt{m/3}}{\sqrt{m}} = \frac{1}{\sqrt{3}} \approx 0.577, \tag{3}$$

and both widths cancel. The approximation replaces each mean norm by its root mean square and drops the bias's factor $(1+1/d)^{1/2}$; in simulation at our widths replacing mean norms by root mean squares changes r by 0.002%, dropping the bias factor changes it by 0.07%, and a single draw of W_0 moves it by up to 0.3%. A heavy-tailed target with unit variance per coordinate can break the approximation. Our case study breaks the third condition, target normalization, on purpose, so that we can move r and measure its effect.

3 EXPERIMENTAL SETUP

Model and data. MelHuBERT (Lin et al., 2023) is a HuBERT-style (Hsu et al., 2021) masked-prediction encoder on log-mel input, with 12 transformer blocks of width 768, each ending in a layer normalization. We train it on LibriSpeech train-clean-100 (Panayotov et al., 2015), about 100 hours in 28,539 utterances, with its published 20 ms configuration: Adam at learning rate 10^{-4} and the loss on masked frames only. We change only the target and, where stated, the readout. Each configuration, which we call an arm, trains for 8,500 steps on two seeds; three arms continue to 12,000.

Targets. We cluster training frames with k-means ($k = 500$) and replace each masked frame's target with an 80-dimensional vector derived from its cluster's centroid. A linear map sets how fast the target's singular spectrum decays, through an exponent γ, and a scalar sets its norm. Arms named `gamma` followed by a number use that exponent, with a scale that comes with it. Two more arms cross shape and scale: `decay2_scale1` has the $\gamma = 2$ shape at the $\gamma = 1$ scale, and `decay1_scale2` the reverse. Two arms zero the readout, giving $r = 0$, on the $\gamma = 1$ and $\gamma = 2$ targets (`zerohead_large` and `zerohead_small`, named for their target's size). Table 1 lists every arm with its r, computed over all 18,124,252 training frames. r ranges from 0 to 37.

Rank. At each block we take the representations of LibriSpeech dev-clean frames and compute the singular values of the frame-by-feature matrix, without subtracting its mean, as RankMe does. Our rank, j^*, counts those at least 1% of the largest. It reads 518 before training and at most 767 in any run. We report block 5 of 12, the middle of the encoder. With the mean kept, a direction shared by every frame counts once however much energy it carries, and a large one pushes the rest below the 1% line. So from every kept checkpoint we also read j^* with the mean over frames removed, and the *mean share* $n\|\bar{h}\|^2/\|H\|_F^2$, with the n frames as the rows of H and $\bar{h}$ their mean, on a fixed set of 120,000 frames (Appendix C). At every block the arms fall in the order of r at step 8,500, with the recovered arms tied at the ceiling, except that the two largest ratios, 35 and 37, swap at most blocks (Appendix C). An arm has *recovered* when j^* first reaches half its ceiling (383.5), interpolating linearly between readings.

Quality. Rank is not quality (Jung et al., 2024), so beside it we report a linear phone classifier trained on frame-level phone labels and scored on held-out speakers, and in the appendix RankMe (Garrido et al., 2023).

4 RESULTS

4.1 THE RATIO ORDERS HOW LONG THE COLLAPSE LASTS

The collapse is fast. Figure 1a shows the first 500 steps for the $\gamma = 1$ and $\gamma = 2$ targets, at $r = 4.5$ and $r = 37$. Both fall from 518 to 26 within twenty steps and reach 18 by step 50. For 200 steps the two curves agree within 1 in j^*, which Section 4.4 explains; most of the fall is a shared offset (Section 4.2).

Every default-readout run collapses, and the ratio orders how long. Every arm with a default readout was below the recovery line at its first reading. Table 1 gives when each came back. At $r = 0.14$ recovery takes 1,064 and 1,345 steps. At $r = 4.5$ to 4.8 it takes 3,530 to 5,071. At $r = 14$ it takes 7,875 on one seed; the other seed was read only at step 8,500, where it sits at 378, just short of the line. At $r = 35$ and 37 neither seed recovers, by 8,500 and 12,000 steps. The zeroed readouts recover before their first reading at 1,000 steps. Some recovery times are bounds, so we count only pairs whose order the readings settle: of the 94 pairs at different ratios among the 15 runs with a recovery reading, 90 recover in the order of r, none against it, and 4 pair the two unrecovered arms, $r = 35$ and 37.

The recovery line does not choose the result. At every recovery line from 30% to 70% of the ceiling, none of the 76 pairs of runs whose ratios differ by more than 2.5 times (any ratio below 0.4 read as 0.4) recovers against the order of r, and at most 2 are undetermined (Appendix C).

Table 1: Every arm, ordered by r. Recovery is the step at which j^* at block 5 first reaches 383.5; $\leq$ marks an arm already past the line at its first reading and $>$ one that never reached it. The two arms at $r = 0$ have a zeroed readout; the rest use PyTorch's default. The second seed of `gamma1.25` was read only at step 8,500. j^* and phone accuracy are at the last step shown.

arm	γ	$\mathbb{E}\|y\|$	r	recovery, seed 0	recovery, seed 1	last step	j^* there	probe there
zerohead_large	1	1.169	0	$\leq$1,000	$\leq$1,000	12,000	767 / 767	0.471 / 0.466
zerohead_small	2	0.142	0	$\leq$1,000	$\leq$1,000	8,500	767 / 767	0.453 / 0.448
gamma0.5	0.5	37.995	0.14	1,064	1,345	8,500	767 / 767	0.441 / 0.443
gamma1	1	1.169	4.52	3,530	4,085	12,000	767 / 767	0.452 / 0.451
decay2_scale1	2	1.099	4.81	4,272	5,071	8,500	767 / 764	0.456 / 0.445
gamma1.25	1.25	0.365	14.47	7,875	–	8,500	415 / 378	0.443 / 0.440
decay1_scale2	1	0.151	35.07	$>$8,500	$>$8,500	8,500	113 / 42	0.407 / 0.359
gamma2	2	0.142	37.32	$>$12,000	$>$12,000	12,000	186 / 127	0.440 / 0.435

Scale, not shape. The γ arms change a target's spectral shape and its scale together, so we separated the two in three ways. `gamma1` and `decay2_scale1` have different shapes at nearly the same ratio, 4.52 and 4.81, and recover together, within 25% of each other seed for seed at every level. `decay1_scale2` has `gamma1`'s shape at $r = 35$ and does not recover by 8,500 steps. And multiplying the $\gamma = 2$ target itself by 7.76, which changes its scale and nothing else, lowers r from 37 to 4.8 and lifts j^* at step 500 from 23 to 123 (Section 4.3).

Quality follows, by less. At step 8,500 the default readout on the $\gamma = 2$ target reaches 0.421 and 0.408 phone accuracy on the two seeds and the zeroed readout 0.453 and 0.448. Across arms, the two still collapsed read 0.36 to 0.42 and the recovered ones 0.44 to 0.48; `gamma1.25`'s second seed, just short of the line at 378, reads 0.440. A short run hides the recovery: `gamma1` reads $j^* = 37$ at step 500 and 767 at step 8,500.

We claim the order, not a curve. With eleven runs, four of them censored, the shape of the dependence is not identified; a power law through the recovered arms misses `gamma2` (Appendix B).

4.2 Most of the collapse is a shared offset

The representation converges on one vector. Before training the mean over frames carries 0.16 of the representation's energy; at $r = 37$ and 4.5 it carries 0.97 by step 20 and 0.98 by step 50 (Figure 4a), so every frame maps to nearly the same vector, which is what a constant predictor needs (Section 4.4). With the mean removed, j^* falls from 563 to 354, a loss of 37% against 97% for raw j^*.

The offset is what r adds. The mean share peaked at 0.30 for a readout scaled to $r = 0.79$ and 0.32 for a softmax readout, and read 0.17 or less for the zeroed readouts at their first reading. Losing rank with the mean removed is not specific to large r: the softmax falls further, to 289 at step 50 (Figure 1a), and the zeroed readout on the larger target reads 355 and 299 at step 500, against 330 at $r = 37$.

The ratio orders both by the end of training. At step 8,500 the mean carries 0.73 to 0.89 of the energy at $r = 35$ and 37, 0.23 and 0.32 at $r = 14$, and at most 0.22 at $r \leq 4.8$ (Table 3). In all 88 pairs of runs whose ratios differ by more than 2.5 times, the smaller r has the smaller mean share and the larger RankMe with the mean removed; on j^* with the mean removed it is larger in 64 pairs and tied at the ceiling in 24. The gap left without the offset is about twofold, 404 and 384 at $r = 37$ against 767 at $r \leq 4.8$, where raw j^* differs sevenfold. Where we kept every checkpoint, rank with the mean removed reaches the ceiling at 3,000 to 4,500 steps with a zeroed readout, at 7,000 and 7,500 at $r = 4.8$, and not by 8,500 at 35 or 37 (Figure 7).

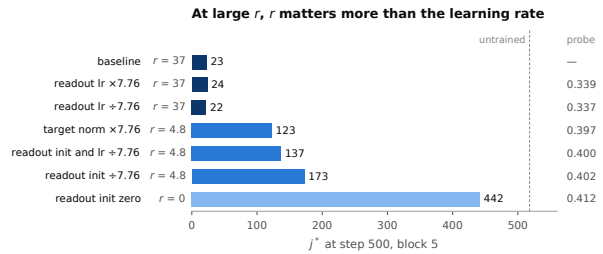


Figure 2: j^* at block 5 after 500 steps on the $\gamma = 2$ target, one seed. The first three rows change only the readout's learning rate at $r = 37$; the rest lower r. Phone accuracy is on the right; the baseline run recorded rank only. Dashed line: the untrained network.

4.3 The ratio matters more than the readout's learning rate

Section 2.2 showed that scaling the target changes the readout's initial size and its effective learning rate together. A smaller readout also moves more easily relative to its own size, so the clock of Section 4.1 could be a clock of mobility. We separated the two in 500-step runs on the $\gamma = 2$ target, on one seed. Before running them we wrote down that raising the readout's learning rate at fixed r would leave j^* at or below 40.

It did (Figure 2). Holding the initialization fixed at $r = 37$ and multiplying the readout's learning rate by 7.76 gives $j^* = 24$. Dividing it by 7.76 gives 22, and the default gives 23. That is a sixty-fold span in learning rate and a change of two in rank, though all three sit near the floor of 18. The faster readout did move 6.0 times further; its mobility changed and the collapse did not. Lowering r to 4.8 instead gives 123 to 173, and zeroing the readout gives 442. At $r = 4.8$ the learning rate does matter, less than r: dividing it by 7.76 lowers 173 to 137. Two of these runs, the enlarged target and the shrunk readout with its learning rate divided, are the same run up to Adam's ϵ by Section 2.2; they read 123 and 137, which indicates how much a single 500-step reading can vary. With the mean removed, where we kept the checkpoints, the default and the faster readout read 330 and 325 and the readout scaled to $r = 0.79$ reads 451. Phone accuracy follows: 0.337 and 0.339 at $r = 37$, and 0.397 to 0.412 when r is lowered. Scaling the readout's initialization alone gives the same picture: j^* at step 500 is 418, 401 and 393 for r of 0.04, 0.37 and 0.79, and 111 at 3.73 (Appendix D).

4.4 What the encoder receives at initialization

The target term is small at large r. The gradient at the encoder's output, the post-layernorm h that enters the readout, is

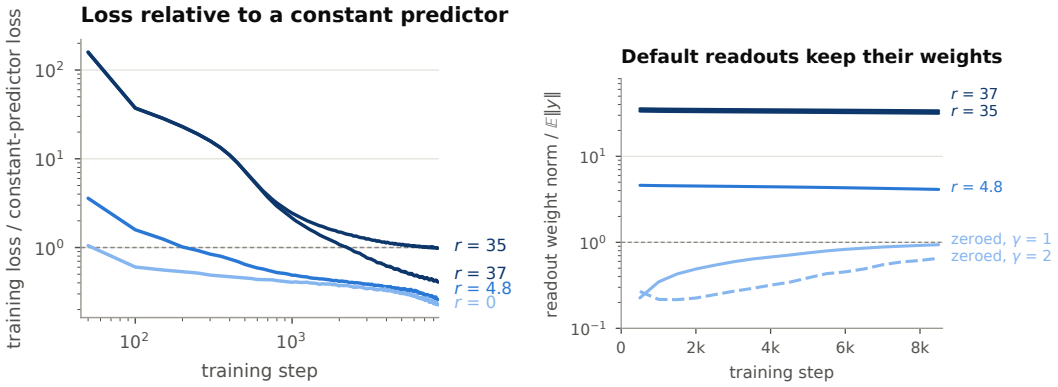
$$\frac{\partial L}{\partial h} = W^\top(Wh + b - y) = W^\top(Wh + b) - W^\top y.$$

For a random W the first term is about r times the second, so at $r = 37$ the target is a 3% correction and $-\partial L/\partial h \approx -W^\top W h$. On real frames through our initial network the target term is 2.6% of the target-free one at $r = 37$, 22% at $r = 4.5$ and 7.0 times larger at $r = 0.14$ (Appendix A.3). At $r = 4.5$ the target term is still a fifth, so both runs of Figure 1a start out driven mostly by the same target-free term. That fits Figure 1a: the two runs part only once the target starts to matter, at 37 against 23 by step 500.

An approximate closed form for its direction. Write $M = W^\top W$. For h isotropic with $\|h\|^2 = d$, $\mathbb{E}_h[h^\top M h] = \operatorname{tr} M$ and $\mathbb{E}_h\|Mh\|^2 = \operatorname{tr} M^2$, whose expectations over W are $m/3$ and $m(m + d - 1/5)/(9d)$. Replacing each by its expectation, which is accurate when both concentrate, and dropping the $-1/5$,

$$\cos\big(-\partial L/\partial h,\ h\big) \ \approx\ -\sqrt{\frac{m}{d+m}}. \tag{4}$$

For $d = 768$ and $m = 80$ this is -0.3071. On synthetic post-layernorm inputs we measure -0.3062, and over ten shapes from $d = 256$ to 3,072 the worst error is 0.68% (Appendix A.3). Scaling W by c multiplies the update by c^2 and leaves its direction alone, so the cosine is the same at every readout



(a) (b)

Figure 3: **(a)** Training loss divided by the loss of the best constant predictor, both axes logarithmic. The $r = 37$ curve is the run with Adam's $\epsilon = 10^{-14}$, which Section 2.2 shows behaves as the default run does. **(b)** The readout's weight norm, $(\|W\|_F^2 + \|b\|^2)^{1/2}$, over the target's norm during training, logarithmic axis. At initialization it equals r for isotropic inputs. Default readouts keep their weights; zeroed readouts grow toward the target's scale.

scale (Figure 6b). The term still depends on the input through h, but not on the target: every input is multiplied by the same matrix, $-W^\top W$. The part of this term along h reaches only the gain and bias of the last layer normalization, whose backward pass removes that direction; the rest, still free of the target, reaches the encoder.

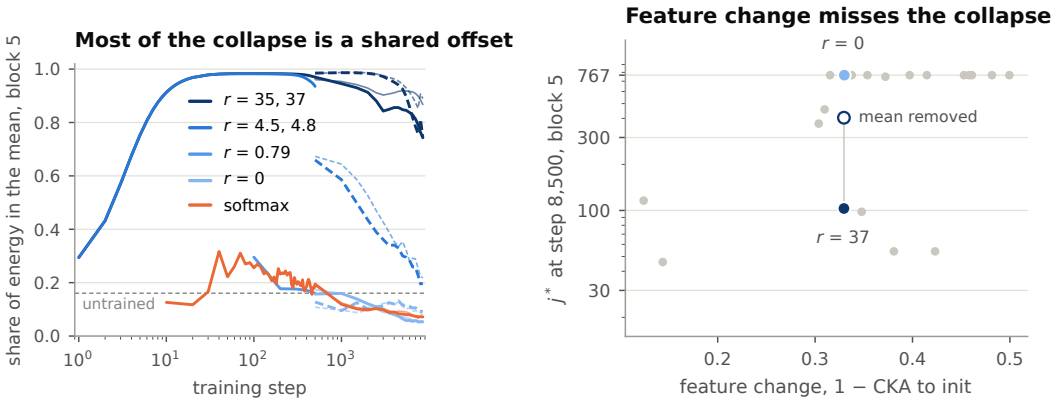
The readout's weights barely shrink. Over 8,500 steps the default readouts keep 0.88 to 0.93 of their initial weight norm (Figure 3b), and a zeroed readout grows until its weight norm is about that of the target. This is not a limit of the optimizer: on the $\gamma = 0.5$ target, where the readout has to grow, it moves 2.03 times its own initial size.

The loss passes through the constant predictor's loss. If the readout's weights stay large, the loss can still fall when the encoder makes the readout's output nearly the same for every input, close to the target's mean: a constant predictor, which a representation dominated by one shared vector can express. Both are measured: the training loss reaches that level (Figure 3a), and the representation is dominated by one shared vector (Section 4.2). This is consistent with the account but does not show that the gradient at initialization causes it. The $r = 37$ run takes 2,250 steps to reach the constant predictor's loss and the zeroed readout 100; at 8,500 steps the zeroed run explains 77.5% of the target's variance and the default one 59.3%, so avoiding the collapse improves the fit.

The loss sets the exponent. Scale the readout by c. Under squared error the gradient at the encoder's output grows as c^2. Under softmax cross-entropy it grows as $O(c)$, because $\partial L/\partial z = p - y$ is bounded. Under a cosine loss it does not grow at all, and with no bias it is orthogonal to h (Appendix A.4). On synthetic post-layernorm inputs the fitted slopes are 2.00, 1.07 and 0.00 (Figure 6a). Training is consistent with this order. A plain softmax readout over the 500 clusters still collapses, but only to $j^* = 202$ at step 40, against a floor of 18 for squared error, and it is back above 300 by step 300 (Figure 1a); a softmax readout ten times larger falls to 26, near squared error's 18 (Appendix E).

4.5 Feature change does not see the collapse

A common way to ask how much training changed a representation is its distance from initialization, for example $1-\text{CKA}$ (Kornblith et al., 2019). It does not register this collapse, and part of the reason is built in: CKA centers each representation before comparing them, so a shared offset is invisible to it. The zeroed and the default readout on the same $\gamma = 2$ target end 0.12% apart on $1 - \text{CKA}$ at seed 0 (0.3300 and 0.3296), with $j^* = 767$ against 103 and phone accuracy 0.430 against 0.414 (Figure 4b). The mean carries 73% of the default run's energy and 5% of the zeroed run's. With the



(a) (b)

Figure 4: **(a)** Share of the representation's energy in its mean over frames. Up to step 500, $r = 37$ and 4.5 are the overlapping runs of Figure 1a; later, full runs (thin: seed 1). **(b)** j^* at step 8,500 against feature change from initialization, twenty arm-seeds. Highlighted: the zeroed and default readouts on one target, seed 0; open circle: the default run with the mean removed.

mean removed the two runs still differ, 767 against 404: centering accounts for most of the sevenfold gap, but a twofold gap remains that CKA also misses. Across twenty arm-seeds the two measures are positively correlated (Spearman 0.48), so feature change is not wrong on average; it cannot see this particular difference. Tomihari & Sato (2024) study readout scale through feature change, so their measurement and ours answer different questions.

5 Published recipes, and how a change can move r

Table 2 computes r for published systems from their papers and code; we did not retrain them. data2vec's readout, an `nn.Linear(768, 768)` at PyTorch's default against a unit-variance target, gives the identity, 0.58, when the eight teacher layers it averages agree, and 1.63 when they are uncorrelated, since averaging shrinks the target's norm. MAE sits at 0.89 because its readout uses Xavier initialization (Glorot & Bengio, 2010) on a 512 to 768 layer. r is undefined for HuBERT and wav2vec 2.0 (Baevski et al., 2020), which score by cosine similarity, and for BEiT (Bao et al., 2022) and BEST-RQ (Chiu et al., 2022), which use a plain softmax; the table gives the protection their loss provides instead.

The regression systems, for which r is defined, sit between 0.58 and 1.63. Our full runs span 0.14 to 37, and our evidence near the recipe is from 500-step runs, where 0.37 and 0.79 kept rank close to a zeroed readout's. A change to the recipe can move r far from there if it (i) regresses an unnormalized target, such as raw features, codec latents or a teacher's unnormalized outputs, (ii) removes the layer normalization in front of the readout, or (iii) initializes the readout at a fixed standard deviation σ, under which $r = \sigma\sqrt{d}$ grows with width. The check costs one batch. If r is far above one, normalizing the target restores the recipe; where the target's scale has to stay, shrinking the readout's initialization by the same factor gives the same r. In our 500-step runs both lifted rank far above the default (Section 4.3).

6 Related work

Readout initialization. Zeroing the final layer is established, in Fixup (Zhang et al., 2019), μP (Yang et al., 2021) and ControlNet (Zhang et al., 2023); we do not propose it as a fix.

Output scale and target scale. Under squared error, multiplying a model's output by α is the same as dividing the target by α, and a large α gives lazy training when the initial output is zero (Chizat et al., 2019); an output multiplier moves a network between lazy and feature training (Geiger et al., 2020). A default readout's initial output is not zero, and when it is large next to the target we

Table 2: r for published systems, computed from their papers and code. The last two rows are Equation 3 and our case study.

system	readout and loss	d	m	r	protection
HuBERT, wav2vec 2.0	cosine, softmax	–	–	–	full: loss ignores readout scale
BEiT, BEST-RQ	linear, softmax	–	–	–	partial: push linear in scale
MAE	linear, squared error	512	768	0.89	normalized target
data2vec, audio	linear, squared error	768	768	0.58–1.63	instance-normed target
standard recipe	linear, squared error	d	m	0.577	the identity, any width
this work, γ=2	linear, squared error	768	80	37.32	none

find the opposite of lazy training: a representation that first collapses. Arnekvist et al. (2020) saw a related effect in MLP regression, where a small target variance makes ReLUs die and the network collapse to a constant function more easily. Pop-Art (van Hasselt et al., 2016) normalizes targets adaptively while preserving the unnormalized outputs. Lyle et al. (2025) show the opposite regime: a large target mean costs plasticity. data2vec normalizes its targets in part because doing so "helps prevent the model from collapsing into a constant representation" (Baevski et al., 2022); we show one way a target's scale can cause such a collapse. Normalized targets, standard in masked image modeling (He et al., 2022; Baevski et al., 2023), thus also fix r.

Readout scale and representations. Closest to us, Masarczyk et al. (2025) show that in softmax classifiers the logits' norm, set by temperature, governs how much rank a network keeps. Theirs is the softmax case of our loss axis and concerns the rank a network ends with; ours is the squared-error case and concerns how long the collapse lasts. In fine-tuning, a randomly initialized head distorts pretrained features (Kumar et al., 2022), and the head's norm controls how far they move (Tomihari & Sato, 2024).

Collapse in self-supervised learning. Dimensional collapse in joint-embedding methods (Jing et al., 2022) motivated variance and covariance regularizers (Bardes et al., 2022) and rank-based evaluation (Garrido et al., 2023). Ours has a different cause, a mismatch of scale between readout and target, and resolves on its own, more slowly at larger r. In joint-embedding methods trained from a small initialization, rank instead rises in steps (Simon et al., 2023), which may describe our recovery but not the collapse before it. Nor is it the loss of rank with depth in pure attention (Dong et al., 2021); ours happens over training steps. Trained representations are known to share a dominant common direction (Ethayarajh, 2019); here a readout's scale drives nearly all the energy into one within fifty steps.

7 Limitations

The training evidence comes from one model family, one corpus and one training budget, with two seeds for full runs and one for the 500-step runs. Equations 1 to 4 and Table 2 are general, but we have not trained those systems, and no full-length run of ours sits near $r = 1/\sqrt{3}$. r is one number: our squared-error readouts were scaled random draws or zero, and readouts with the same output norm but a different spectrum or alignment need not behave alike. `decay1_scale2`'s target is barely learned (1.9% and -2.0% of its variance explained), a confound beside its r. The gradient results of Section 4.4 hold at initialization, up to a target term of relative size about $1/r$ and the concentration behind Equation 4; they explain how the collapse starts, not how long it lasts. Rank with the mean removed and the mean share come only from the checkpoints we kept, and early in training they separate the arms far less than raw rank does.

8 Conclusion

Behind a layer normalization, a regression target's size sets r, and in our case study r ordered how long the encoder stayed collapsed, mostly onto one shared vector. The standard recipe keeps r near $1/\sqrt{3}$ at every width; a change to the target, the normalization or the initialization can move it far from there, and one forward pass shows whether it has.

AI USE STATEMENT

We used Claude (Anthropic) to brainstorm ideas and improve the clarity and presentation of the manuscript. All AI-assisted work was reviewed and verified by the authors, who take full responsibility for the final content.

REPRODUCIBILITY STATEMENT

The supplementary material contains the code, every recorded measurement, and a command that rebuilds every number, table and figure in this paper from those measurements on a CPU in about two minutes. Appendix B gives the training setup and every change we made to MelHuBERT; Appendix A derives Equations 1 to 4. Trained weights are not included because of their size.

REFERENCES

Isac Arnekvist, J. Frederico Carvalho, Danica Kragic, and Johannes A. Stork. The effect of target normalization and momentum on dying ReLU. *arXiv preprint arXiv:2005.06195*, 2020.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020. arXiv:2006.11477.

Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. data2vec: A general framework for self-supervised learning in speech, vision and language. In *Proceedings of the 39th International Conference on Machine Learning (ICML 2022)*, volume 162 of *Proceedings of Machine Learning Research*, pp. 1298–1312, 2022. arXiv:2202.03555.

Alexei Baevski, Arun Babu, Wei-Ning Hsu, and Michael Auli. Efficient self-supervised learning with contextualized target representations for vision, speech and language. In *Proceedings of the 40th International Conference on Machine Learning (ICML 2023)*, 2023. arXiv:2212.07525.

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT pre-training of image transformers. In *The Tenth International Conference on Learning Representations (ICLR 2022)*, 2022. arXiv:2106.08254.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Chung-Cheng Chiu, James Qin, Yu Zhang, Jiahui Yu, and Yonghui Wu. Self-supervised learning with random-projection quantizer for speech recognition. In *Proceedings of the 39th International Conference on Machine Learning (ICML 2022)*, volume 162 of *Proceedings of Machine Learning Research*, pp. 3915–3924, 2022. arXiv:2202.01855.

Lénaïc Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International Conference on Machine Learning (ICML)*, 2021.

Kawin Ethayarajh. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 55–65, 2019. doi: 10.18653/v1/D19-1006.

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. In *Proceedings of the 41st International Conference on Machine Learning (ICML 2024)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 12666–12700, 2024. arXiv:2407.05872.

Quentin Garrido, Randall Balestriero, Laurent Najman, and Yann LeCun. RankMe: Assessing the downstream performance of pretrained self-supervised representations by their rank. In *Proceedings of the 40th International Conference on Machine Learning (ICML 2023)*, volume 202 of *Proceedings of Machine Learning Research*, 2023. arXiv:2210.02885.

Mario Geiger, Stefano Spigler, Arthur Jacot, and Matthieu Wyart. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020 (11):113301, 2020. doi: 10.1088/1742-5468/abc4de.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256, 2010.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2022)*, pp. 16000–16009, 2022. arXiv:2111.06377.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. HuBERT: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3451–3460, 2021. doi: 10.1109/TASLP.2021.3122291. arXiv:2106.07447.

Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Euna Jung, Jaeill Kim, Jungmin Ko, Jinwoo Park, and Wonjong Rhee. Unveiling key aspects of fine-tuning in sentence embeddings: A representation rank analysis. *arXiv preprint arXiv:2405.11297*, 2024.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR 2015)*, 2015. arXiv:1412.6980.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529, 2019. arXiv:1905.00414.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations (ICLR 2022)*, 2022. arXiv:2202.10054; ICLR 2022 Oral.

Tzu-Quan Lin, Hung-yi Lee, and Hao Tang. MelHuBERT: A simplified HuBERT on mel spectrograms. In *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2023. doi: 10.1109/ASRU57964.2023.10389700. arXiv:2211.09944.

Clare Lyle, Zeyu Zheng, Khimya Khetarpal, Hado van Hasselt, Razvan Pascanu, James Martens, and Will Dabney. Disentangling the causes of plasticity loss in neural networks. In *Proceedings of The 3rd Conference on Lifelong Learning Agents*, volume 274 of *Proceedings of Machine Learning Research*, pp. 750–783, 2025.

Wojciech Masarczyk, Mateusz Ostaszewski, Tin Sum Cheng, Tomasz Trzciński, Aurelien Lucchi, and Razvan Pascanu. Unpacking softmax: How temperature drives representation collapse, compression, and generalization. *arXiv preprint arXiv:2506.01562*, 2025.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. LibriSpeech: An ASR corpus based on public domain audio books. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015. doi: 10.1109/icassp.2015.7178964.

James B. Simon, Maksis Knutins, Liu Ziyin, Daniel Geisz, Abraham J. Fetterman, and Joshua Albrecht. On the stepwise nature of self-supervised learning. In *International Conference on Machine Learning (ICML)*, 2023.

Akiyoshi Tomihari and Issei Sato. Understanding linear probing then fine-tuning language models from NTK perspective. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, 2024. doi: 10.52202/079017-4436. arXiv:2405.16747.

Hado van Hasselt, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. Learning values across many orders of magnitude. In *Advances in Neural Information Processing Systems 29*, 2016.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10524–10533, 2020. arXiv:2002.04745.

Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs V: Tuning large neural networks via zero-shot hyperparameter transfer. In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*, 2021. arXiv:2203.03466.

Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. In *International Conference on Learning Representations (ICLR 2019)*, 2019. arXiv:1901.09321.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV 2023)*, pp. 3813–3824, 2023. doi: 10.1109/ICCV51070.2023.00355. arXiv:2302.05543.

A Derivations

Throughout, $h \in \mathbb{R}^d$ is one position's representation entering the readout, with $\|h\|^2 = d$ (exact up to the layer normalization's ϵ); the readout is W ($m \times d$) and b (m), every entry drawn independently from $\mathcal{U}(-1/\sqrt{d}, 1/\sqrt{d})$, so each has mean zero and variance $s = 1/(3d)$; the loss on a position is $L = \frac{1}{2}\|Wh + b - y\|^2$.

A.1 The readout's first output

Row i gives $(Wh)_i = \sum_j W_{ij} h_j$, a sum of independent zero-mean terms, so $\mathbb{E}(Wh)_i^2 = s\|h\|^2 = 1/3$. Summing over the m rows and adding the independent bias, whose coordinates have variance s,

$$\mathbb{E}\,\|W_0 h + b_0\|^2 = \frac{m}{3} + \frac{m}{3d} = \frac{m}{3}\Big(1 + \frac{1}{d}\Big).$$

Nothing here involves y. The same computation prices other initializers. Xavier uniform has variance $2/(d+m)$, so $\mathbb{E}\|Wh\|^2 = 2dm/(d+m)$; for MAE's readout ($d = 512$, $m = 768$) against a normalized target this gives $r = 0.894$. A fixed standard deviation σ gives $\mathbb{E}\|Wh\|^2 = md\sigma^2$, so $r = \sigma\sqrt{d}$ against a unit-variance target, which grows with width.

A.2 Target scale as readout scale

Let run A train on $s\,y$ with readout (W, b) and run B on y with readout $(W/s, b/s)$, with the same encoder, the same data order and Adam's state starting at zero in both. Their residuals are $\rho_A = Wh + b - sy$ and $\rho_B = \rho_A/s$. Then $L_B = L_A/s^2$; the gradient with respect to the encoder is $W^\top \rho$ times the encoder's Jacobian, so it is $1/s^2$ as large in run B; and the gradient with respect to W is $\rho h^\top$, $1/s$ as large. Adam's update is invariant to a constant rescaling of a parameter's gradient, up to its ϵ, so the encoder receives the same update in both runs. The readout's parameters in run B are $1/s$ the size of run A's, so matching their trajectory requires run B's readout learning rate to be divided by s. A target's scale is therefore a statement about two properties of the readout: its initial size and its learning rate. Section 4.3 varies them separately.

A.3 The direction of the gradient

At large r, $-\partial L/\partial h \approx -Mh$ with $M = W^\top W$, and $\cos(-Mh, h) = -h^\top Mh/(\|Mh\|\,\|h\|)$. Take h isotropic with $\|h\|^2 = d$, so that $\mathbb{E}[h^\top Mh] = \operatorname{tr} M$ and $\mathbb{E}\|Mh\|^2 = \operatorname{tr} M^2$. The trace $\operatorname{tr} M$ is the sum of all md squared entries, $m/3$. With w_i the rows of W, $\operatorname{tr} M^2 = \sum_{i,k}(w_i \cdot w_k)^2$. The m diagonal terms are $\mathbb{E}\|w_i\|^4$ and the $m(m-1)$ off-diagonal ones are $\mathbb{E}(w_i \cdot w_k)^2 = ds^2$; with the fourth moment of a uniform variable, $\mathbb{E}\,w^4 = 1/(5d^2)$, these sum to $m(m + d - 1/5)/(9d)$. To leading order in $1/d$,

$$\cos \approx -\frac{m/3}{\sqrt{d \cdot m(m+d-1/5)/(9d)}} = -\sqrt{\frac{m}{m+d-1/5}} \approx -\sqrt{\frac{m}{d+m}}.$$

Replacing W by cW multiplies Mh by c^2 and leaves the cosine unchanged. Figure 5 compares the closed form with simulation over ten shapes; the worst error is 0.68%. On real frames, passed through our initial network with each frame's target drawn from the arm's target distribution, the target term is 2.6% of the target-free term at $r = 37$, 22% at $r = 4.5$ and 7.0 times larger at $r = 0.14$; the full gradient's cosine with the target-free term is 0.9996, 0.976 and 0.146. Its cosine with h at large r is -0.32 rather than -0.307, because real representations are not isotropic.

A.4 Why the loss sets the exponent

Scale the readout by c. Under squared error, $W^\top(Wh + b - y) \approx W^\top Wh$ grows as c^2. Under softmax cross-entropy the gradient is $W^\top(p - y)$ and $\|p - y\| \le \sqrt{2}$ whatever c is, so it grows at most as c; a large random readout is confidently wrong, $p - y$ sits at its bound, and the growth is linear. Under a cosine loss the loss depends only on $x/\|x\|$ with $x = Wh$. Its gradient in x is orthogonal to x and scales as $1/c$, which cancels the c in $W^\top$, so the push does not grow; with no bias, $h^\top \partial L/\partial h = x^\top \partial L/\partial x = 0$, so it is exactly orthogonal to h. The loss decides how much of

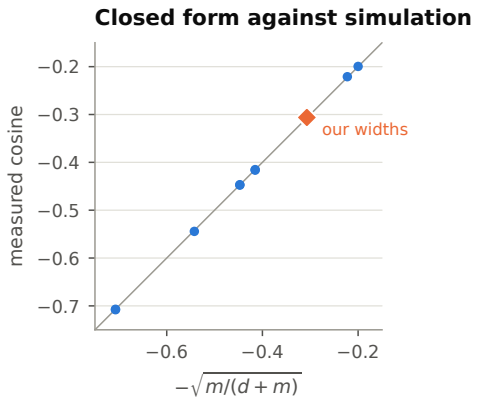
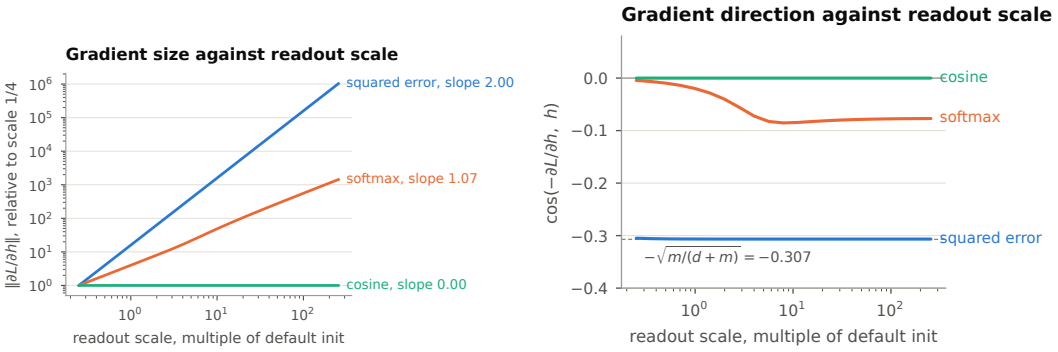


Figure 5: Equation 4 against the simulated cosine on synthetic post-layernorm inputs, for ten combinations of d and m. The highlighted point is our widths, $d = 768$ and $m = 80$, with the target and the bias included.

the answer the readout's magnitude has to supply: all of it under squared error, part of it under a softmax, none of it under a cosine.



(a) (b)

Figure 6: At initialization, with the readout held fixed, on synthetic post-layernorm inputs. **(a)** Size of the gradient at the encoder's output against readout scale, for three losses, both axes logarithmic. **(b)** Its cosine with h, readout scale logarithmic. The dashed line is Equation 4.

A.5 How far the readouts move

Adam's step on a parameter is at most about its learning rate, so over 8,500 steps an entry could move about 0.85, some forty times a typical initial entry. No readout comes near that budget, so it does not explain why the large ones stay large. The softmax cells of Appendix E, whose readouts differ 100-fold in norm, move 4.26 and 0.014 of their own norm in 500 steps. On the $\gamma = 0.5$ target, where the readout must grow, it moves 2.03 times its own size in 8,500 steps. The readouts that stay put are the ones that would have to shrink; the encoder absorbs the mismatch instead.

A.6 The constant predictor

The best constant prediction is the target's mean over masked positions, and its loss is the target's variance. We call that loss the null; $1 - \text{loss}/\text{null}$ is the share of the target's variance a model explains. An encoder that maps every position to the same representation can reach the null and no lower.

B SETUP

Model. MelHuBERT at commit `49f2346c`: 12 transformer blocks of width 768, each closing with a layer normalization, over log-mel input at 20 ms per step. The readout is one linear layer, $768 \to 80$ for dense targets and $768 \to 500$ for the softmax arm, at PyTorch's default initialization. Training uses MelHuBERT's 20 ms configuration: Adam with learning rate 10^{-4}, $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, no weight decay, gradient clipping at norm 10 (which never fired), 8 gradient-accumulation steps, the loss on masked frames only.

What we changed in MelHuBERT. Current NumPy and PyTorch compatibility fixes; dense targets loaded as floats, with squared-error loss when the objective is set to it; a second Adam parameter group for the readout with its own learning-rate multiplier; optional encoder freezing for a number of steps; optional linear warmup; and saving a checkpoint every given number of steps. Each change after the first does nothing unless enabled, so a default run is the upstream code.

Data. Training on LibriSpeech train-clean-100: 28,539 utterances, 18,124,252 frames. Evaluation on LibriSpeech dev-clean with frame-level phone labels from forced alignments. The phone classifier is linear, trained for 500 steps, and scored on 10 held-out speakers.

Targets. k-means with $k = 500$ on training frames gives a codebook. A frame's target is its cluster's centroid passed through a linear map that sets the spectral exponent γ of the codebook, then scaled. r uses a forward pass over 120,000 frames at initialization for the numerator and every training frame, in float64, for the denominator.

Two probe frame sets. The 500-step runs and the full runs trained the phone classifier on different frame sets, which gives them different majority-class baselines (0.0744 and 0.0748). We never compare phone accuracies across the two, and every table reports one set only.

Statistics. Recovery is interpolated linearly between readings. An arm past the line at its first reading gives an upper bound, and one that never reaches it a lower bound, so each run's recovery time is an interval. A pair of runs at different ratios is in order when the interval of the smaller ratio lies wholly before the other's, against the order when it lies wholly after, and undetermined otherwise; we report counts, not a p-value. The censored fit maximizes the likelihood of $\log T = a + b \log r$ with Gaussian noise, a recovered run contributing its density and an unrecovered one the probability of exceeding its bound. Over the eleven runs with $r > 0$ it gives an exponent of 0.42, with a 95% profile interval of [0.37, 0.50] that holds only if the power law is right; `gamma2`'s bound needs 0.44, inside that interval, so the data neither establish nor exclude a power law. Dropping the worse seed of `decay1_scale2`, whose target the model barely learns, moves the exponent from 0.4242 to 0.4234. The phone classifier starts from zero and trains full-batch, so it is deterministic given the representation.

Compute. One GPU with 24 GB of memory. A training step takes about 1.6 s, so an 8,500-step arm takes about four hours on one seed.

C FULL RUNS

Figure 8 shows every full run at block 5. Table 4 repeats the order test at five recovery levels; the only reversals are between 4.52 and 4.81, and between a zeroed readout and 0.14. Figure 9 shows every block at step 8,500. RankMe, which uses no threshold, orders the arms with default readouts by r at step 8,500 except within the two nearly equal pairs, 4.52 and 4.81, and 35 and 37.

With the mean removed. We read every kept checkpoint once more at block 5, on a fixed set of 120,000 dev-clean frames, the set Section 4.5 uses: every step of the first 500 for the runs of Figure 1a, every 500 steps for five full runs, and the last step for the rest. On this set raw j^* differs from Table 1's by up to 43 at one run (`gamma1.25`, seed 0: 458 against 415), so we never compare the two sets directly. Table 3 gives every full run at step 8,500, and Figure 7 the runs read every 500 steps.

Table 3: Every full run at step 8,500, block 5, on the 120,000-frame set, seed 0 / seed 1; raw j^* differs slightly from Table 1, which uses another frame set. Mean share is the fraction of the representation's energy carried by its mean over frames. Phone accuracy is from the same checkpoints, on the probe frames of Table 1.

arm	r	j^*	j^*, mean removed	RankMe, mean removed	mean share	probe
zerohead_large	0	767 / 767	767 / 767	568 / 567	0.09 / 0.08	0.477 / 0.472
zerohead_small	0	767 / 767	767 / 767	482 / 480	0.05 / 0.06	0.453 / 0.448
gamma0.5	0.14	767 / 767	767 / 767	611 / 611	0.08 / 0.09	0.441 / 0.443
gamma1	4.52	767 / 767	767 / 767	415 / 430	0.13 / 0.16	0.448 / 0.443
decay2_scale1	4.81	767 / 749	767 / 767	449 / 439	0.20 / 0.22	0.456 / 0.445
gamma1.25	14.47	458 / 370	577 / 543	329 / 337	0.23 / 0.32	0.443 / 0.440
decay1_scale2	35.07	116 / 46	486 / 255	322 / 258	0.74 / 0.89	0.407 / 0.359
gamma2	37.32	103 / 54	404 / 384	304 / 293	0.73 / 0.87	0.421 / 0.408

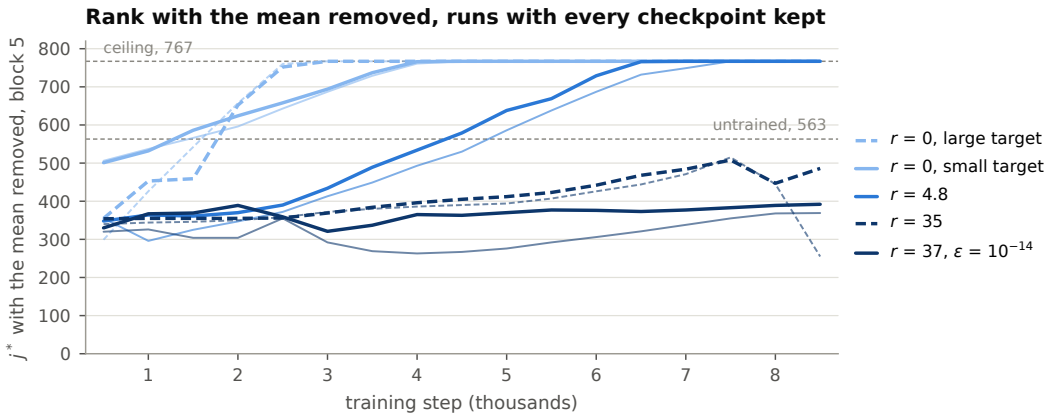


Figure 7: j^* with the mean over frames removed, block 5, for the five full runs whose every checkpoint we kept; thin lines are seed 1.

D THE 500-STEP RUNS

Table 5 lists every 500-step run on one seed. Warmup delays the collapse and does not prevent it, and a longer ramp helps more by step 1,500 (Figure 10). Changing the readout or the encoder after the collapse has started does not help: zeroing the readout at step 100 or freezing the encoder for its first 100 or 300 steps leaves rank lower at step 500 than doing nothing. The frozen runs read raw $j^* = 1$ because the mean carries 0.999816 and 0.999996 of their energy; with it removed they read 227 and 77. The fix belongs at initialization.

E SOFTMAX READOUTS

We trained four softmax readouts for 500 steps: a random or a phone-probe-aligned readout, at 0.1 or 10 times the default size (Figure 11). Size ordered the damage in both rows: the larger random readout fell to 26 and the smaller to 204; the larger aligned readout fell to 74 and the smaller to 171. An alignment installed at initialization more than halves within ten steps, because the encoder moves that fast (Figure 12), and the readout that stayed more aligned is the one that fell further. Magnitude, not alignment, orders the damage.

F PREDICTIONS WRITTEN BEFORE THE RUNS

Before several rounds of runs we wrote down what we expected and the reading that would show we were wrong. Table 6 lists every one with what came back: nine predictions, of which five were

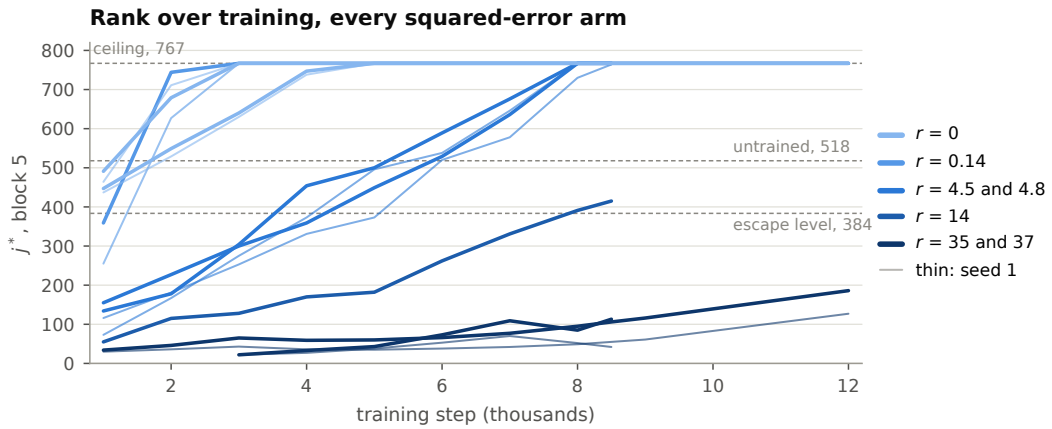
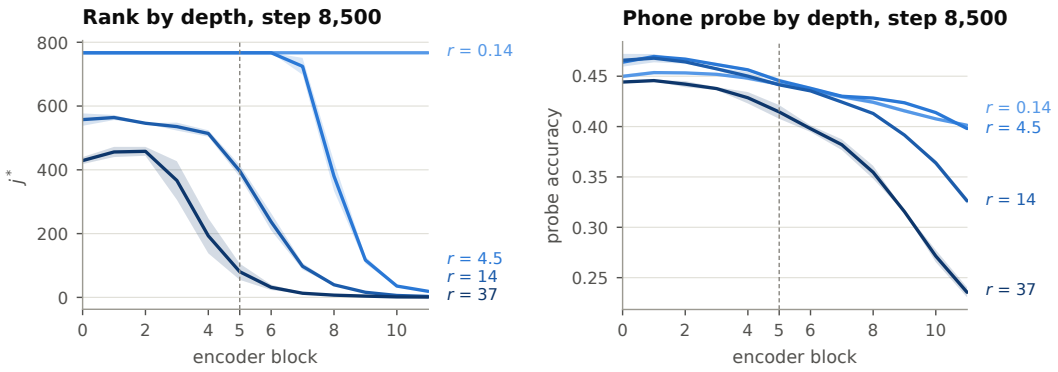


Figure 8: j^* at block 5 over training for every arm; thin lines are seed 1.

Table 4: The order of recovery at five recovery levels, over pairs of runs at different ratios. A pair is against the order when the run with the larger r recovers first, and undetermined when neither run's recovery interval lies wholly before the other's. The last column counts pairs whose ratios differ by more than a factor of 2.5, reading any ratio below 0.4 as 0.4.

share of ceiling	j^* level	pairs: in order / against / undetermined	the same, r apart $> 2.5\times$
30%	230	80 / 2 / 12	76 / 0 / 0
40%	307	85 / 1 / 8	76 / 0 / 0
50%	384	90 / 0 / 4	76 / 0 / 0
60%	460	86 / 0 / 8	74 / 0 / 2
70%	537	82 / 4 / 8	74 / 0 / 2

refuted, and five quicker screens, of which three were refuted. The refutations shaped the paper: we first expected r to act as a threshold, and the clock of Section 4.1 is what we found instead.

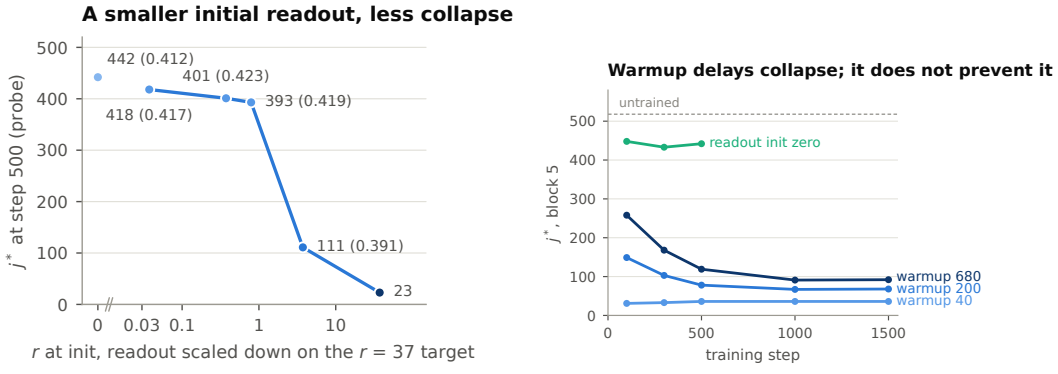


(a) (b)

Figure 9: Every block at step 8,500, mean over seeds with the range shaded. **(a)** j^*. **(b)** Phone accuracy.

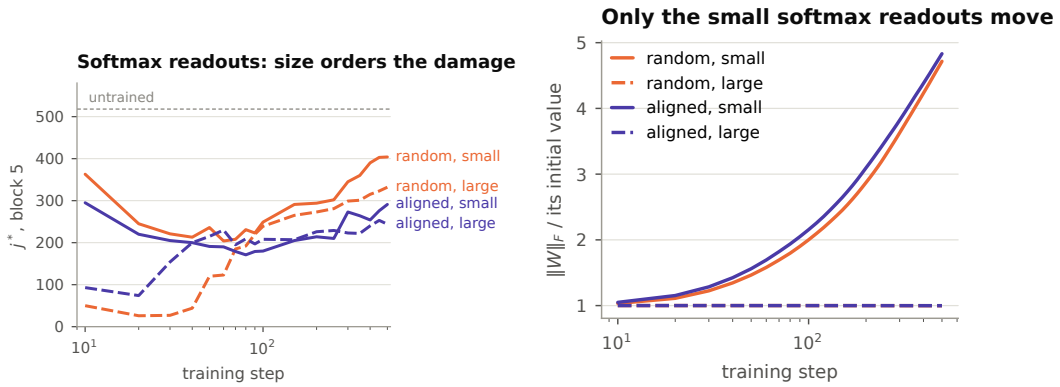
Table 5: Every 500-step run, block 5, one seed, $\gamma = 2$ target unless stated.

group	arm	what changed	r	j^*	probe	RankMe
r against mobility	baseline	nothing	37.32	23	–	66.3
r against mobility	lr_up	readout lr $\times$7.76	37.32	24	0.339	70.3
r against mobility	lr_only	readout lr $\div$7.76	37.32	22	0.337	64.6
r against mobility	target_rescaled	target norm $\times$7.76	4.81	123	0.397	203.8
r against mobility	equiv	readout init and lr $\div$7.76	4.81	137	0.400	216.8
r against mobility	init_only	readout init $\div$7.76	4.81	173	0.402	237.3
r against mobility	zero	readout init zero	0	442	0.412	332.7
r against mobility	zero_gamma1	readout init zero, γ=1 target	0	348	0.426	250.1
readout scale	dose1e-3	readout init $\times$0.001	0.04	418	0.417	329.0
readout scale	dose1e-2	readout init $\times$0.01	0.37	401	0.423	318.9
readout scale	dose2e-2	readout init $\times$0.0212	0.79	393	0.419	313.3
readout scale	dose1e-1	readout init $\times$0.1	3.73	111	0.391	195.9
warmup	warmup40	40-step ramp	37.32	36	0.353	105.0
warmup	warmup200	200-step ramp	37.32	78	0.377	185.4
warmup	warmup680	680-step ramp	37.32	119	0.383	226.6
repair, one seed	late_zero	readout zeroed at step 100	37.32	5	0.290	12.5
repair, one seed	freeze100	encoder frozen to step 100	37.32	1	0.164	2.4
repair, one seed	freeze300	encoder frozen to step 300	37.32	1	0.074	1.1



(a) (b)

Figure 10: **(a)** j^* at step 500 against the readout's initial scale, phone accuracy in brackets. **(b)** Linear warmup of three lengths.



(a) (b)

Figure 11: Four softmax readouts over 500 steps; dashed lines are the larger readouts. **(a)** j^* at block 5. **(b)** The readout's norm relative to its start.

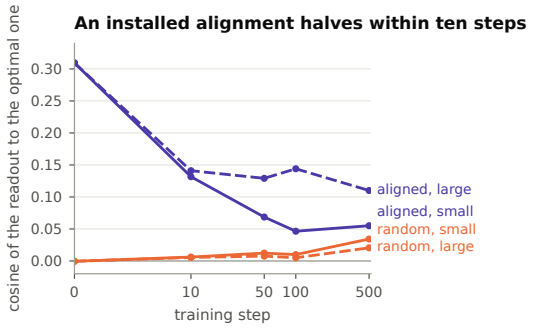


Figure 12: Cosine between each softmax readout and the optimal readout refitted for the encoder at that step.

Table 6: Every prediction, as written before its run, with the reading and the verdict.

prediction	written before the run	reading	verdict
12a	zeroed readout at 12,000 steps keeps a phone-probe margin in [0.005, 0.020]	0.0186 and 0.0151	held
12b	a softmax readout never falls below j^* 260 in 500 steps	202 at step 40	refuted
12c	freezing the encoder at step 100 lands under j^* 200	1	held, for a reason that was wrong
12d	freezing the encoder at step 300 stays at or above j^* 400	1	refuted
15a	among random softmax readouts, the larger one falls further	26 against 204	held
15b	among aligned softmax readouts, the larger one falls less	74 against 171	refuted
15c	the two scale effects have opposite signs	-178 and -97	refuted
15d	no softmax cell falls below j^* 50	26	refuted
lr_up	raising the readout's learning rate 7.8x at fixed r leaves j^* at or below 40	24	held
screen A	warmup delays the collapse and does not prevent it	all three ramps collapse	held
screen B	zeroing the readout at step 100 recovers partway	5, below doing nothing	refuted
screen C	j^* falls monotonically with readout scale	442, 418, 401, 393, 111, 23	held
screen C	the cliff sits where r = 1	393 at r 0.79, written as r = 1 under an early target-norm estimate; the cliff lies in (0.79, 3.73]	refuted
screen 9	a longer warmup is worse	$36 < 68 < 92$ at step 1,500: longer is better	refuted