WHAT MAKES THE WEIGHTS OF IMPLICIT NEURAL REPRESENTATIONS CLASSIFIABLE?

Anonymous authors
Paper under double-blind review

ABSTRACT

Implicit neural representations (INRs) store an image in the parameters of a small network, and recent work classifies images by reading these parameters. The Meta Weight Transformer (MWT) made such classifiers far more accurate by learning, jointly with a Transformer classifier, how sinusoidal INRs (SIRENs) are fitted. Earlier work relates accuracy to measures of class structure in the parameters, but such measures show what can be read from the parameters, not what a classifier uses. In this paper, we study whether class structure reflects how well MWT classifies, and which inputs its classifier relies on. We first compare nearest-neighbor and nearest-centroid readouts of this structure with the trained classifiers across six MWT training configurations. The readouts are far less accurate and do not rank the configurations by accuracy. Next, we change the inputs of the trained classifiers without retraining them, and find that the classifiers depend heavily on the per-neuron biases, which make up less than 0.4% of their input values. Replacing the biases by their layer means lowers accuracy by 16.3 points on average, whereas the same change to a weight column of the same size lowers it by only 0.3. This dependence centers on a low-dimensional variation of the biases across images, one that closely matches coarse color and brightness statistics. Our results show that, in MWT, the fitted parameters are classifiable in part because their biases carry coarse information about the image.

1 INTRODUCTION

Implicit neural representations (INRs) have recently emerged as a new way to represent images. Instead of a grid of pixels, an INR stores an image as a small neural network that maps each pixel coordinate to its color. Among INRs, sinusoidal representation networks (SIRENs) are a common choice, because their sine activations let them reconstruct fine image detail such as sharp edges and textures (Sitzmann et al., 2020). Since the network is a continuous function of the coordinates, the representation is not tied to the resolution of a pixel grid. Building on this flexibility, a growing line of work uses the INR itself as the input to downstream tasks (Dupont et al., 2022), classifying an image by reading the parameters of its network instead of its pixels.

Reading these parameters is, however, not straightforward. Many different parameter settings represent the same image. For example, the hidden neurons of a SIREN can be permuted without changing its output (Ainsworth et al., 2023). Classifiers designed to be equivariant to such symmetries (Navon et al., 2023; Zhou et al., 2023a;b) still fall far behind classifiers that read pixels. Papa et al. (2024) train the classifier after fitting and find that fitting all images from the same initialization makes the parameters easier to classify. They also find that the better a clustering of the parameters agrees with the labels, the more accurate the classifier. The Meta Weight Transformer (MWT; Gielisse & van Gemert, 2025) instead learns the shared initialization and the step sizes of a few fitting steps together with a Transformer classifier, which reads each neuron's incoming weights and bias as one token. Because the classification loss is back-propagated through the fitting steps, how each image is fitted depends on the classifier as well as on the reconstruction objective. This joint training made SIREN classifiers far more accurate. MWT itself, however, does not examine what structure the joint training gives the parameters. A measure such as clustering agreement shows what can be read from the parameters, but work on probing learned representations has found that such information can go unused by the model (Ravichander et al., 2021; Elazar et al., 2021). It therefore remains unclear

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.

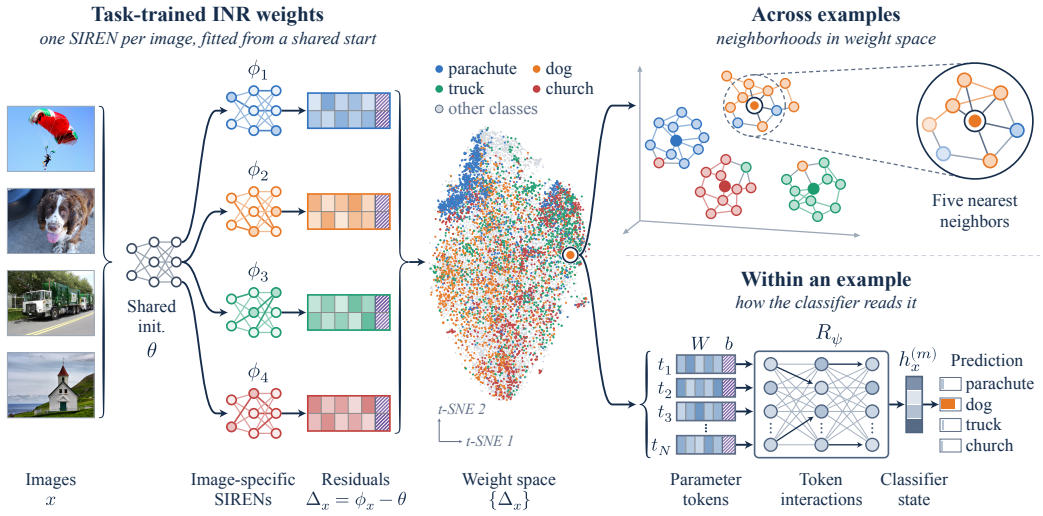

Figure 1. Two views of the fitted parameters. Each image is fitted as a SIREN from a shared initialization θ. The fitted parameters ϕ_x differ from θ by an image-specific residual Δ_x. Across examples (top right; schematic), we measure class structure among these residuals, for example by five-nearest-neighbor voting (Section 4.2). Within an example (bottom right; schematic), a trained Transformer classifier R_ψ reads the residuals as tokens, each holding a neuron's incoming weights W and its bias b. We change these tokens to find which inputs it depends on (Sections 4.3 and 4.4). Its internal state after block m is denoted $h_x^{(m)}$ (read out in Figure 2C). The center is a t-SNE of the baseline residuals of all 13,394 Imagenette images.

whether class structure in the fitted parameters reflects how well MWT classifies, and which parts of the parameters its classifier relies on.

In this paper, we study these questions in MWT. Across six training configurations that differ in accuracy by about four points, we classify each image by its nearest neighbors or by its nearest class centroid among the fitted parameters, and compare these readouts of class structure with the trained classifiers. The readouts are far less accurate and do not track which configurations classify better. Inside the classifier, however, nearest-neighbor accuracy rises from block to block, and after the final block both readouts rank the configurations closely by accuracy.

Next, we intervene on the inputs of the trained classifiers to find which parts of the parameters they rely on (Figure 1). A fitting step changes each bias by an error signal summed over the sampled pixels, but changes each weight by that signal multiplied by the output of the previous sine layer, which oscillates across the image (Section 3.3). Given this difference, we focus on the per-neuron biases. They make up less than 0.4% of the classifier's input but can carry information about the image as a whole, such as its overall color and brightness. Without refitting the SIRENs or retraining the classifiers, we replace the biases by their layer means, set them to zero, or permute them across images, and make the same change to a weight column of the same size. The classifiers turn out to depend heavily on the biases and on which image they come from. To understand what the biases record about the image, we predict them from color and brightness statistics and remove the predicted part. Finally, we compare four of the configurations with the baseline on Imagenette and CIFAR-10. Three change how the classifier reads the parameters, and one adds an objective that pulls same-class parameters together. Our main contributions can be summarized as follows:

- We show that simple readouts of class structure in the fitted parameters reach at most 42%, compared with 58–63% for the trained classifiers, and that the nearest-neighbor and nearest-centroid readouts do not rank the configurations by accuracy.
- We design paired interventions on the classifiers' inputs and find that replacing the biases by their layer means lowers accuracy by 16.3 points on average, compared with 0.3 for a

weight column of the same size. The classifiers depend on a low-dimensional variation of the biases that closely matches coarse color and brightness statistics.

- We find that configurations that change the classifier are up to 6.9 points more accurate than the baseline on Imagenette and CIFAR-10, whereas an objective that pulls same-class parameters together lowers accuracy.

2 Related Work

Classification of INRs and network weights. Early work used network weights as inputs to predict properties of trained models, such as their accuracy or hyperparameters (Unterthiner et al., 2021; Schürholt et al., 2021). For INRs, classifiers that read the weights have been built to respect the permutation symmetry of hidden neurons, with equivariant layers, Transformers, or graph networks over neurons (Navon et al., 2023; Zhou et al., 2023a;b; Kofinas et al., 2024). Other work gives the classifier a more compact input than the full weights, either per-example latents that shift the biases of a shared network (Dupont et al., 2022; Bauer et al., 2023) or an embedding of the fitted weights (De Luigi et al., 2023). Papa et al. (2024) and MWT (Gielisse & van Gemert, 2025) instead keep the full weights and change how they are fitted. Janik & Byra (2026) combine MWT's joint fitting with a hierarchical mixture-of-experts classifier and report higher accuracy. On the biases themselves, Zhang et al. (2025) find that in SIRENs fitted to single images, the biases matter for reconstruction mainly in the input layer and bias updates add little. They also store features of a pretrained network in frozen input-layer biases and use these biases for classification. By contrast, the classifier in MWT reads the hidden-layer biases fitted to each image, and we study how it uses them.

Probing and intervention-based analyses. A probe, usually a linear classifier, measures what can be decoded from a representation (Alain & Bengio, 2017). Because decodable information need not be what a model uses (Ravichander et al., 2021; Belinkov, 2022), intervention-based analyses remove information from a representation and measure how the model's behavior changes (Elazar et al., 2021), often by erasing a concept linearly (Ravfogel et al., 2020; Belrose et al., 2023). For classifiers trained as in MWT, Janik & Byra (2026) score each input parameter by gradient-based attribution. We instead change whole groups of classifier inputs, such as all biases, and compare each change with the same change on an equal number of weights.

Class structure in learned representations. Center losses and supervised contrastive learning pull same-class representations together (Wen et al., 2016; Khosla et al., 2020), and neural collapse describes the simple class geometry that emerges near the output of supervised networks (Papyan et al., 2020). These works concern the features a network computes, whereas the fitted parameters in MWT are the input of its classifier. We measure class structure both in these parameters and at the classifier's final block (Section 3.2), and one configuration applies a center loss to them (Section 3.4).

3 Method

We study the fitted parameters from the two views of Figure 1. Across examples, we measure how much class structure can be decoded from them (Section 3.2). Within an example, we change a trained classifier's inputs to see what it uses (Section 3.3). Both analyses use six training configurations of MWT that differ in accuracy (Section 3.4).

3.1 Image fitting and classifier inputs

For an image x, a SIREN predicts RGB values from two-dimensional pixel coordinates. MWT fits every image from a shared initialization θ in k steps with learned per-parameter step sizes α. The fitted parameters and their residual are

$$\phi_x = U_\alpha^k(\theta; x), \qquad \Delta_x = \phi_x - \theta. \tag{1}$$

We use $k = 4$, as MWT does on Imagenette. Each step reduces the reconstruction error on randomly sampled pixels. During training, the classification loss is back-propagated through these steps. The initialization, the step sizes, and the classifier are therefore learned together (Gielisse & van Gemert, 2025), as in the meta-learning methods MAML and Meta-SGD (Finn et al., 2017; Li et al., 2017). The fitting steps themselves use only the pixels of an image, not its label.

Table 1. Class readouts of the fitted parameters and of the final classifier block. Accuracy (%) on Imagenette, mean over three seeds (s.d. in Table B1). 5-NN and centroid readouts use the training images as references, whereas linear and MLP readouts are trained on 80% of the validation images and tested on the rest. Last row: Spearman correlation with classifier Top-1 across the six configurations (Section 3.4). Chance is 10%.

		Fitted parameters				Final block	
Configuration	Top-1	5-NN	Centroid	Linear	MLP	5-NN	Centroid
● Baseline	60.7	19.9	41.0	39.8	37.7	55.8	58.1
● Composite	62.3	20.7	41.9	40.4	41.2	58.9	59.6
● Composite + denser fitting	62.6	21.5	41.6	40.6	40.3	60.4	60.8
● Bias encoder	62.3	20.7	41.0	39.8	38.9	59.4	60.2
■ Center auxiliary	59.0	19.2	41.1	38.8	36.3	54.0	55.4
■ Instance-contrastive auxiliary	58.3	21.5	41.7	40.1	38.8	53.4	56.1
Spearman ρ with Top-1		0.14	0.09	0.66	0.77	0.94	0.89

The classifier reads the four hidden-to-hidden weight matrices of the SIREN and their biases, which make up nearly all of the SIREN's parameters. Each neuron becomes one token that contains the residuals of its incoming weights and of its bias. MWT adds to each token a learned shift β, the same for every image, and scales the result by a fixed factor λ:

$$t_{x,\ell,j} = \lambda\left([\Delta W_{x,\ell,j},\, \Delta b_{x,\ell,j}] + \beta_{\ell,j}\right), \tag{2}$$

where ℓ indexes the layer and j the neuron. With 256 neurons per hidden layer (width 256), this gives a $1{,}024 \times 257$ input matrix whose last column holds the biases, less than 0.4% of the input values. The baseline classifier maps each token to 256 dimensions with an input projection, applies ten Transformer blocks, averages over tokens, and applies a linear head.

3.2 Measuring class structure in the fitted parameters

We compare each readout of the fitted parameters with the trained classifiers in accuracy (Table 1) and in how it ranks the six configurations (Spearman correlation with classifier Top-1). All readouts use the residual Δ_x reduced to 128 principal components. Because all images are fitted from the same θ, their residuals can be compared entry by entry. Five-nearest-neighbor (5-NN) voting and nearest-centroid classification, with the training images as references, ask whether an image lies closer to its own class than to others. The centroid margin $(d_x^- - d_x^+)/(d_x^- + d_x^+)$ measures how much closer an image lies to its own class centroid than to the nearest other one, where d_x^+ and d_x^- are the distances to these two centroids. Linear and two-layer MLP readouts instead learn a class boundary from 80% of the validation images. A classifier of the baseline architecture trained afresh on the frozen parameters shows how much it can find in them without joint training (Appendix B.3). Finally, to see whether the 5-NN and centroid readouts track accuracy once class structure is learned, we apply them to the output of the classifier's final block, averaged over tokens as in the baseline.

3.3 Intervening on the inputs of trained classifiers

To find which inputs a trained classifier depends on, we replace a group of entries of its input matrix (for example, all biases) by their layer means, set them to zero, or permute them across images, and measure the change in Top-1 accuracy. A layer mean is taken over all images and neurons of a SIREN layer. The fitted SIRENs and the classifier's parameters stay as trained.

Each operation on the biases is paired with a second one (Table 2). A drop in accuracy might follow from changing any 1,024 input values rather than from the biases in particular. As a control, we apply the same three operations to one weight column in each SIREN layer, and these columns together also hold 1,024 values. In each layer, we take the column whose spread is closest to that of the biases and call it the selected weight column. In other pairs, both operations act on the biases. One removes either the fitted residual, which differs between images, or the learned shift β, which all images share. Another permutes the biases across all images or only within a class.

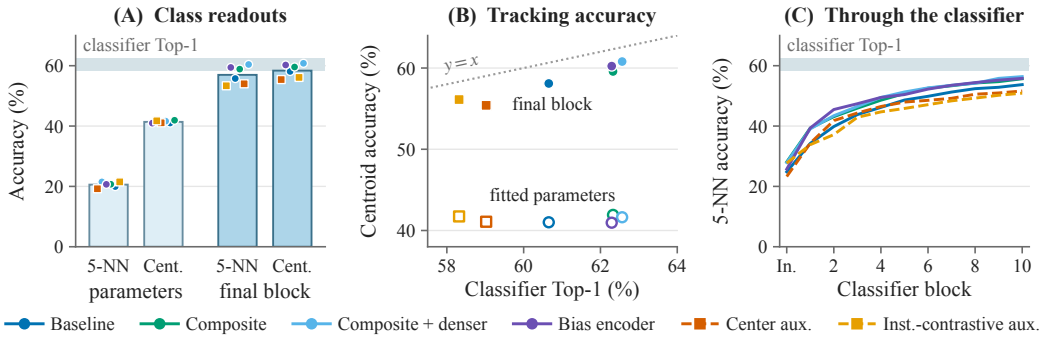

Figure 2. Class structure in the fitted parameters and across the classifier's blocks. Colors mark configurations (squares and dashed lines: auxiliary objectives); the shaded band spans classifier Top-1. (A) The 5-NN and centroid readouts of Table 1 for the fitted parameters (light bars) and the final block (dark bars); bars: mean over configurations; dots: configurations (mean over seeds). (B) Centroid readouts against classifier Top-1: open markers for the fitted parameters, filled for the final block. (C) 5-NN after the input projection (In.) and each block, with leave-one-out voting among the validation images rather than the training references of Table 1; mean over three seeds.

Biases and weights differ in how their residuals are fitted. Let $\delta_j(p)$ be the gradient of the reconstruction loss $\mathcal{L}$ with respect to the pre-activation of neuron j at a sampled pixel p, and $a_i(p)$ the input from neuron i of the previous layer. A fitting step changes each parameter in proportion to its gradient, and the gradients of the bias b_j and the weight W_{ji} are

$$\partial\mathcal{L}/\partial b_j = \textstyle\sum_p \delta_j(p), \qquad \partial\mathcal{L}/\partial W_{ji} = \textstyle\sum_p \delta_j(p)\, a_i(p). \tag{3}$$

In a SIREN, $a_i(p)$ is the output of a sine and oscillates across the image, whereas the bias gradient lacks this factor. The bias residuals may therefore reflect coarse properties of the whole image.

To test this, we predict the biases linearly from color and brightness statistics. Next, we remove the predicted part, a linear erasure related to those of Ravfogel et al. (2020) and Belrose et al. (2023), to see whether the classifiers use it. For comparison, we also remove about as much variance along the biases' leading principal components, computed without these statistics (Appendix D).

3.4 Training configurations

Besides the baseline, we train five configurations. Three change the classifier, and two add an auxiliary objective on the residuals. Composite changes both the classifier and the fitting feedback. It combines global attention with attention among the tokens of neighboring SIREN layers, pools the tokens of each SIREN layer separately, and, over training, increases how strongly the classification loss acts on the fitting. Composite + denser fitting also samples twice as many pixels per fitting step. Bias encoder uses the attention of Composite with the pooling and fitting feedback of the baseline, and passes each bias through a small learned encoder before the input projection. Center auxiliary adds to the training objective a term that pulls each residual toward the mean direction of its class, and otherwise matches the baseline. Instance-contrastive auxiliary fits two augmented views of each image and pulls the embeddings of their residuals together without using labels (Chen et al., 2020). Appendix A gives the full definitions and training settings.

4 Experiments

4.1 Setting

We run three sets of experiments: readouts of class structure in the fitted parameters (Section 4.2), interventions on the inputs of the trained classifiers (Sections 4.3 and 4.4), and a comparison of complete training configurations (Section 4.5). The first two use Imagenette (Howard, 2019), a ten-class subset of ImageNet (Deng et al., 2009) with 9,469 training and 3,925 validation images at 320×320, and SIRENs of width 256. The comparison and the layer-mean replacement of Section 4.3

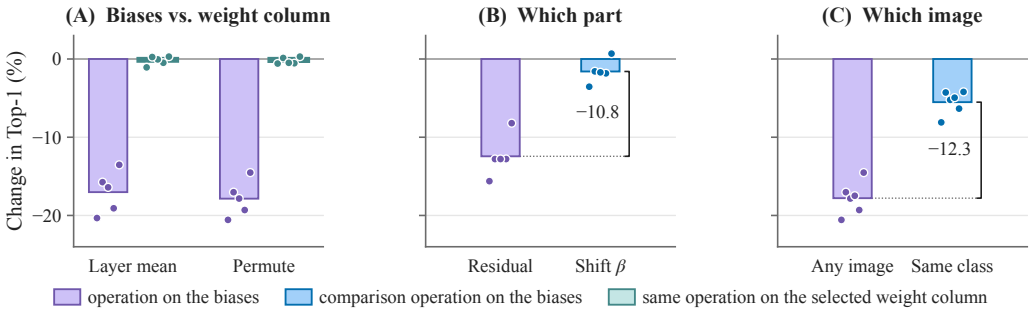

Figure 3. Operations on the biases of trained classifiers. Change in Top-1 accuracy in percentage points, one seed per configuration; bars: mean over configurations (Table 2); dots: individual configurations; brackets: mean paired difference. (A) Layer means and permutation, on the biases and on the selected weight column (Section 3.3). (B) Removing the fitted residual (violet) or the learned shift β (blue). (C) Permutation among all images (violet) or only within a class (blue).

also use CIFAR-10 (Krizhevsky, 2009), with SIRENs of width 128 and 256. We train each of the six configurations of Section 3.4 with three seeds. Accuracy is the Top-1 of the checkpoint selected on validation data. The baseline reproduces the width-256 model of MWT, which Gielisse & van Gemert (2025) call MWT-L (60.7% on Imagenette, compared with the reported 60.8%). The interventions and the classifiers trained afresh in Section 4.2 use one seed per configuration unless stated otherwise. Because Bias encoder passes the biases through a separate encoder, no weight column of its input can serve as their control. We use it only for permuting the biases and for zeroing them one layer at a time. The other interventions use the remaining five configurations.

4.2 What readouts of the parameters capture

Readouts of the fitted parameters are far less accurate than the trained classifiers (Table 1, Figure 2A). The 5-NN readout reaches about 20% and the other readouts at most 42%, compared with 58–63% for the classifiers. A classifier of the baseline architecture trained afresh on the frozen parameters of each configuration reaches 51–54% (Appendix B.3). The parameters therefore hold more class information than these readouts expose.

The 5-NN and centroid readouts do not rank the configurations by accuracy (Spearman $\rho = 0.14$ and 0.09; Table 1, Figure 2B). The linear and MLP readouts partly follow classifier accuracy ($\rho = 0.66$ and 0.77). All four readouts of the parameters in Table 1 place Instance-contrastive auxiliary above the baseline, but its classifier is less accurate (58.3% vs. 60.7%). The centroid margin ranks the configurations closely by accuracy ($\rho = 0.94$), although its average is close to zero in every configuration (Appendix B.2). Class structure among the fitted parameters thus follows accuracy when measured by the centroid margin, not by the nearest centroid.

Applied to the output of the classifier's final block, the 5-NN and centroid readouts rank the configurations closely by accuracy ($\rho = 0.94$ and 0.89; Table 1, Figure 2A–B). Measured after each block with leave-one-out voting among the validation images (Figure 2C), 5-NN accuracy rises from 23–28% at the input projection to 51–56% after the final block.

4.3 Which inputs the classifiers depend on

The classifiers depend strongly on the biases, far more than on the selected weight column (Figure 3A). Over five configurations and three seeds, replacing the biases by their layer means costs 16.3 points on average, compared with 0.3 points for the same operation on the selected weight column. The gap holds on every checkpoint (Figure 4B; Appendix C.3). In four of the five configurations, the selected column has a 1.4 to 5.9 times smaller spread than the biases. In the fifth, Center auxiliary, the two spreads are similar, but the gap remains (15.1 vs. 0.2 points over three seeds; Appendix C.1). Setting the biases to zero costs about as much as replacing them, while setting the selected weight column or sixteen randomly chosen weight columns to zero costs less than one point (Table 2, Figure 4A). On

Table 2. Paired interventions on trained classifiers. Change in Top-1 accuracy in percentage points, one seed per configuration, mean ± s.d. over five configurations, or six for permutations of the biases. Each column pairs an operation on the biases with the same operation on the selected weight column (left) or with a second operation on the biases (right). The last row gives the mean paired difference.

	Biases vs. weight column			Biases vs. biases		
	Layer mean	Zero	Permute	Residual / shift β	Any image / same class	Color / PCs
Operation	−17.0 ±2.7	−16.5 ±2.8	−17.8 ±2.1	−12.4 ±2.7	−17.8 ±2.1	−14.7 ±2.0
Comparison	−0.2 ±0.6	−0.4 ±0.4	−0.2 ±0.4	−1.6 ±1.5	−5.5 ±1.5	−14.9 ±2.0
Difference	−16.8	−16.1	−17.6	−10.8	−12.3	+0.2

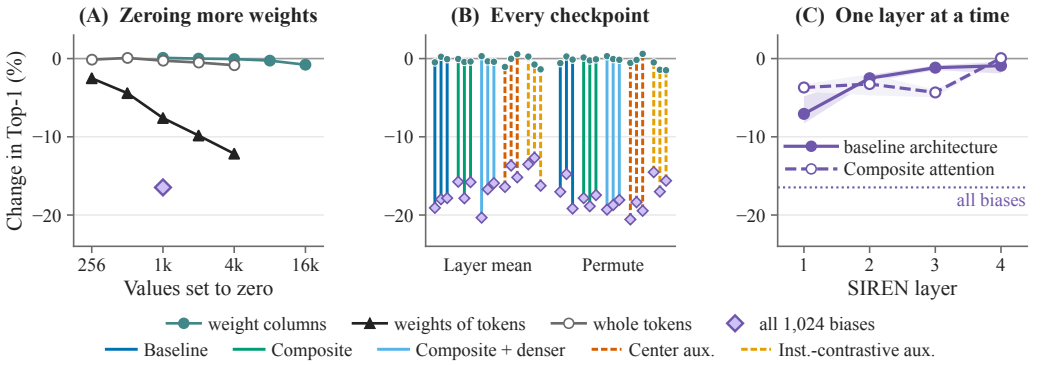

Figure 4. Size, consistency, and location of the bias effect. Change in Top-1 accuracy in percentage points. (A) Values set to zero: weight columns, the weights of tokens with their biases kept, or whole tokens; one seed per configuration, mean over five configurations. (B) Layer means and permutation on each of the 15 checkpoints, for the biases (diamonds) and the selected weight column (circles). (C) Biases of one SIREN layer set to zero, mean over three seeds; mean and range over the configurations of each classifier (baseline architecture: Baseline, Center aux., Inst.-contrastive aux.; Composite attention: Composite, Composite + denser, Bias encoder); dotted: all biases set to zero (one seed).

CIFAR-10, replacing the biases by their layer means costs 8.7 and 14.6 points on average at widths 256 and 128, compared with 0.1 and 1.9 points for the selected weight column (Appendix C.4).

The classifiers depend on which image the biases come from, not only on whether the bias values resemble those seen in training. Permuting the bias vectors across images keeps their distribution and the learned shift β, yet it costs 17.8 points, about as much as replacing them by their layer means. The same permutation of the selected weight column costs 0.2 (Figure 3A). Removing the fitted residual leaves only the shared shift β and costs 12.4 points, whereas removing β costs 1.6 (Figure 3B).

The biases also carry information about the class of their image, and the classifiers use it. Permuting the biases only among images of the same class costs 5.5 points, a third as much as permuting them across all images (Figure 3C). After a permutation across all images, predictions also move toward the class of the image that supplied the biases. When an image receives the biases of an image from another class, it is predicted as that class in 11.9% of cases, compared with 6.5% for the class of an unrelated image drawn by a second permutation (all 18 checkpoints; Appendix C.2).

A permutation, however, also pairs each bias with the weights of another image. To check whether this mismatch alone disrupts the classifiers, we set the weights of four random tokens to zero but keep their biases. This changes 1,024 values, as the column controls do, and costs 7.6 points on average (Appendix C.2), whereas zeroing the same tokens entirely costs 0.3 (Figure 4A). A bias without its own weights thus disrupts the classifiers by itself. This mismatch can explain part of the cost of permutation, but not the shift of predictions toward the class of the image that supplied the biases.

Zeroing the biases of one SIREN layer at a time shows where the dependence lies (Figure 4C). For classifiers of the baseline architecture, the first layer costs the most, up to 8.3 points (mean over three seeds). For those with Composite attention, the loss spreads over the first three layers. Even Bias

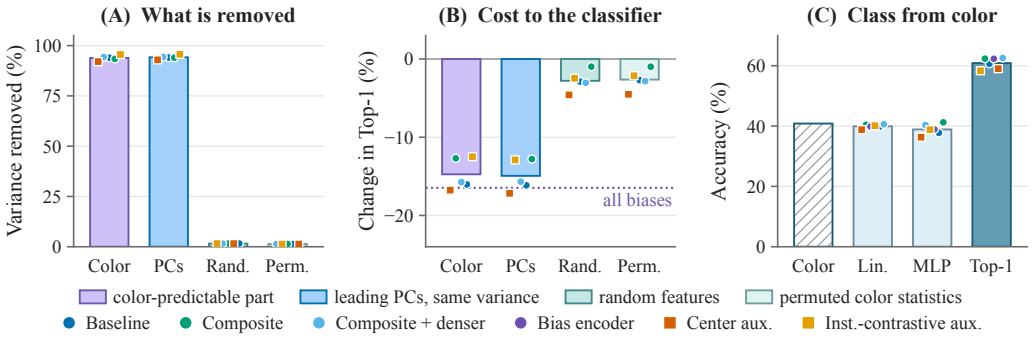

Figure 5. What varies in the biases, and what removing it costs. Bars: mean over configurations; dots: configurations. (A, B) One seed of each of five configurations. We remove from the biases either the part predicted by 58 features (color statistics, random, or permuted color statistics) or about as much variance along the leading principal components. (A) Share of bias variance removed. (B) Change in Top-1 accuracy in percentage points; dotted: all biases set to zero. (C) Linear readout of the color statistics alone (Color, hatched; one value), linear and MLP readouts of the fitted parameters (Lin., MLP), and classifier Top-1; six configurations, three-seed means.

Table 3. Top-1 of the modified configurations in three settings. Top-1 (%) is the mean ± s.d. over three seeds, and Δ the difference from the baseline mean in points, computed before rounding. Network width is given in parentheses, and CIFAR-10 uses 32 × 32 images. Per-seed values and a plot of the differences are in Appendix E.

	Imagenette (256)		CIFAR-10 (128)		CIFAR-10 (256)	
Configuration	Top-1	Δ	Top-1	Δ	Top-1	Δ
● Baseline	60.65 ±0.83	–	57.44 ±1.84	–	62.16 ±0.05	–
■ Center auxiliary	59.02 ±0.55	−1.63	55.79 ±0.25	−1.65	58.39 ±0.11	−3.76
● Composite	62.33 ±0.21	+1.67	59.18 ±0.04	+1.75	61.84 ±0.46	−0.32
● Bias encoder	62.29 ±0.49	+1.64	60.70 ±0.29	+3.26	63.88 ±0.21	+1.73
● Composite + denser fitting	62.56 ±0.69	+1.91	64.31 ±0.07	+6.88	66.80 ±0.59	+4.64

encoder, which encodes the biases separately, loses 18.5 points when they are permuted across images (three seeds). Changing the classifier moves the dependence across layers but does not remove it.

4.4 What varies in the biases

The biases differ between images mainly along a few directions; the four leading principal components carry 93–96% of the bias variance. As Equation 3 suggests, this variation reflects coarse statistics of the image. A ridge regression on 58 color and brightness statistics, such as the mean and spread of each color channel over the image and its quadrants, accounts for 92–96% of the total variance of the biases, whereas the same regression on 58 random or permuted features accounts for less than 2% (Figure 5A; Appendix D). A linear readout of these 58 statistics alone, on a separate validation split, reaches 40.8%, as high as the linear readouts of the fitted parameters (Figure 5C).

Removing the part of the biases predicted by the color statistics costs 14.7 points, close to the 16.5 points lost when all biases are set to zero. The same removal with random features costs 2.8 points (Figure 5B, Table 2). Removing about as much variance along the leading principal components, computed without the color statistics, costs 14.9 points. The classifiers therefore depend on this dominant variation, which the color statistics track closely.

4.5 Which configurations classify better

Center auxiliary adds an objective that pulls same-class parameters together. It is 1.6 to 3.8 points less accurate than the baseline in all three settings of Table 3. On Imagenette, its auxiliary loss falls

to nearly zero, but not because the classes separate. For all three seeds, its class mean directions are nearly parallel (pairwise cosine similarity above 0.999), and its readouts of the fitted parameters in Table 1 stay close to the baseline's (Appendix E). The configurations that change the classifier are more accurate than the baseline in eight of their nine comparisons, by 1.6 to 6.9 points, and the gain is largest for Composite + denser fitting, which also samples twice as many pixels.

5 Discussion

Although we study MWT with SIRENs on Imagenette and CIFAR-10, replacing the biases by their layer means needs no retraining and can be applied to other classifiers of INR weights, such as equivariant ones, to measure how much of their accuracy rests on the biases. Recolored test images would show whether MWT's classifiers respond to color and brightness as such, and whether this limits their robustness, as reliance on simple cues does for pixel classifiers (Geirhos et al., 2020).

6 Conclusion

In this paper, we studied what makes the weights of implicit neural representations classifiable when their fitting is trained together with a classifier, using MWT as a case study. We found that simple readouts of class structure in the fitted parameters expose only part of their class information, and that nearest-neighbor and nearest-centroid readouts do not track which configurations classify better. To see what the classifiers use, we designed paired interventions on their inputs. Although the per-neuron biases make up less than 0.4% of the input values, the classifiers rely heavily on them and little on a weight column of the same size. They depend on a low-dimensional variation of the biases, and coarse color and brightness statistics of the image track this variation closely. Part of what makes the fitted parameters classifiable in MWT is the coarse image information that their biases carry. Finally, on Imagenette and CIFAR-10, an objective that pulls same-class parameters together lowers accuracy, whereas configurations that change the classifier are up to 6.9 points more accurate than the baseline.

AI Use Statement

In this work, we used generative AI tools for translation and for implementing parts of the code of our methods. We have not used generative AI tools for proposing or refining hypotheses, designing experiments, cleaning or organizing datasets, generating synthetic data, or interpreting results, and developing theoretical models or conceptual frameworks, formulating or proving mathematical claims, and qualitative or thematic analysis are not applicable to this work. Additionally, we used generative AI tools to polish the wording of parts of the paper, to explain key concepts, and to check parts of the code and text. We have reviewed all AI-assisted work: the authors checked and tested the AI-implemented code and verified all translations and edits. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility Statement

The supplementary material contains anonymized code, which builds on the public implementation of MWT, for training the six configurations and for the interventions of Section 3.3 and Appendices C and D. Imagenette (Howard, 2019) and CIFAR-10 (Krizhevsky, 2009) are publicly available, and Appendix A.1 describes the preprocessing and augmentation. The baseline reproduces the width-256 model of MWT (Section 4.1). Appendix A defines the six configurations and gives their training settings (Tables A1 and A2). Each configuration is trained with three seeds, and accuracy is the Top-1 of the checkpoint selected on validation data (Appendix A.3). Appendix B gives the protocols of the readouts and of the classifiers trained afresh, Appendix C.1 defines the interventions, Appendix D defines the color statistics and the ridge removal, and Appendix E gives the Top-1 of every seed.

References

Samuel Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git Re-Basin: Merging models modulo permutation symmetries. In *International Conference on Learning Representations*, 2023.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. In *International Conference on Learning Representations (Workshop Track)*, 2017.

Matthias Bauer, Emilien Dupont, Andy Brock, Dan Rosenbaum, Jonathan Richard Schwarz, and Hyunjik Kim. Spatial functa: Scaling functa to ImageNet classification and generation. *arXiv preprint arXiv:2302.03130*, 2023.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.

Nora Belrose, David Schneider-Joseph, Shauli Ravfogel, Ryan Cotterell, Edward Raff, and Stella Biderman. LEACE: Perfect linear concept erasure in closed form. In *Advances in Neural Information Processing Systems*, 2023.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, 2020.

Luca De Luigi, Adriano Cardace, Riccardo Spezialetti, Pierluigi Zama Ramirez, Samuele Salti, and Luigi di Stefano. Deep learning on implicit neural representations of shapes. In *International Conference on Learning Representations*, 2023.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

Emilien Dupont, Hyunjik Kim, S. M. Ali Eslami, Danilo Jimenez Rezende, and Dan Rosenbaum. From data to functa: Your data point is a function and you can treat it like one. In *International Conference on Machine Learning*, 2022.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Transactions of the Association for Computational Linguistics*, 9:160–175, 2021.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

Alexander Gielisse and Jan van Gemert. End-to-end implicit neural representations for classification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Jeremy Howard. Imagenette: A smaller subset of 10 easily classified classes from ImageNet. https://github.com/fastai/imagenette, 2019.

Stanislaw Janik and Michal Byra. Weight-space mixture-of-experts for implicit neural representation classification. *arXiv preprint arXiv:2607.29463*, 2026.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, 2020.

Miltiadis Kofinas, Boris Knyazev, Yan Zhang, Yunlu Chen, Gertjan J. Burghouts, Efstratios Gavves, Cees G. Snoek, and David Zhang. Graph neural networks for learning equivariant representations of neural networks. In *International Conference on Learning Representations*, 2024.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-SGD: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

Aviv Navon, Aviv Shamsian, Idan Achituve, Ethan Fetaya, Gal Chechik, and Haggai Maron. Equivariant architectures for learning in deep weight spaces. In *International Conference on Machine Learning*, 2023.

Samuele Papa, Riccardo Valperga, David Knigge, Miltiadis Kofinas, Phillip Lippe, Jan-Jakob Sonke, and Efstratios Gavves. How to train neural field representations: A comprehensive study and benchmark. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Vardan Papyan, X. Y. Han, and David L. Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40): 24652–24663, 2020.

Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. Null it out: Guarding protected attributes by iterative nullspace projection. In *Annual Meeting of the Association for Computational Linguistics*, 2020.

Abhilasha Ravichander, Yonatan Belinkov, and Eduard Hovy. Probing the probing paradigm: Does probing accuracy entail task relevance? In *Conference of the European Chapter of the Association for Computational Linguistics*, 2021.

Konstantin Schürholt, Dimche Kostadinov, and Damian Borth. Self-supervised representation learning on neural network weights for model characteristic prediction. In *Advances in Neural Information Processing Systems*, 2021.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems*, 2020.

Thomas Unterthiner, Daniel Keysers, Sylvain Gelly, Olivier Bousquet, and Ilya Tolstikhin. Predicting neural network accuracy from weights. *arXiv preprint arXiv:2002.11448*, 2021.

Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, 2016.

Weixiang Zhang, Boxi Li, Shuzhao Xie, Chengwei Ren, Yuan Xue, and Zhi Wang. Understanding bias terms in neural representations. In *Advances in Neural Information Processing Systems*, 2025.

Allan Zhou, Kaien Yang, Kaylee Burns, Adriano Cardace, Yiding Jiang, Samuel Sokota, J. Zico Kolter, and Chelsea Finn. Permutation equivariant neural functionals. In *Advances in Neural Information Processing Systems*, 2023a.

Allan Zhou, Kaien Yang, Yiding Jiang, Kaylee Burns, Winnie Xu, Samuel Sokota, J. Zico Kolter, and Chelsea Finn. Neural functional transformers. In *Advances in Neural Information Processing Systems*, 2023b.

A TRAINING AND IMPLEMENTATION

A.1 SHARED FITTING PROCEDURE

Imagenette has 9,469 training and 3,925 validation images, resized to 320×320, with RGB values mapped to $[-1, 1]$. Training images are augmented with reflection padding, rotations of up to 30 degrees, cropping, resized cropping with scale 0.8–1.0, and horizontal flips; validation images are only resized. CIFAR-10 images are 32×32.

The SIREN has a coordinate-input layer, four square hidden-to-hidden layers, and an RGB output layer with a learned output scale. The coordinate-input layer uses frequency factor 30 and the later sine layers factor 1. The output layer and its scale are shared by all images and are not fitted per image. At width 256, the fitted residual has 263,936 entries: 768 in the coordinate-input layer and 263,168 in the four square layers with their biases. Only the latter enter the classifier as tokens; the *full residual* below means all 263,936 entries.

The residual starts at zero. Each of the four fitting steps samples a fraction of the pixel coordinates (Table A2), computes the squared reconstruction error on them, and updates the residual with the learned per-parameter step sizes α (Li et al., 2017). After the last step, a reconstruction loss on sampled pixels enters the training objective. The classification loss is back-propagated through the fitting steps; its gradients reaching the shared SIREN parameters and the step sizes are multiplied by the feedback coefficient (Table A2), and those reaching the classifier are not.

Table A1. Training settings of the six configurations (Imagenette). Settings that differ are in Table A2.

	Setting	Value
Model	SIREN width	256
	SIREN layers read	4
	Classifier embedding dimension	256
	Transformer blocks	10
	Token scale λ	500
	Fitting steps (training and evaluation)	4
Optimization	Optimizer	AdamW
	Weight decay	10^{-4}
	Learning rate (shared SIREN and classifier)	10^{-4}
	Learning rate (step sizes α)	10^{-2}
	Scheduler	OneCycleLR
	Warmup fraction	0.05
	Gradient-norm clipping (per parameter group)	1.0
Training	Batch size	16
	Epochs	80
	Seeds	3

A.2 CONFIGURATION DEFINITIONS

Baseline. Each token is linearly projected from 257 to 256 dimensions. Ten pre-normalized attention and feed-forward blocks operate on the token sequence, with residual LayerScale initialized at 0.1. Attention uses head dimension 64. The final representation is the mean over tokens, followed by a linear class head.

Composite. Each attention block combines attention to tokens from neighboring SIREN layers with global attention over all tokens. A learned scalar weights the global contribution. Learned SIREN-layer positional embeddings are added after the input projection. At the output, the tokens are grouped by their four SIREN layers, and the four group means are combined with learned softmax weights. The feedback coefficient increases linearly from 0.001 to 0.1 over training.

Composite + denser fitting. Composite with the fraction of pixels sampled per fitting step raised to 0.10.

Bias encoder. Each bias passes through a shared Linear(1, 16)–GELU encoder, whose 16 outputs are concatenated with the token's 256 weight entries before a Linear(272, 256) input projection. The attention and positional embeddings are those of Composite; the pooling is the baseline's mean over tokens, and the feedback coefficient is fixed at 0.01.

Center auxiliary. Adds to the training objective, with coefficient 1.0, one minus the cosine similarity between each normalized full residual and a running center of its class, averaged over the batch; the centers are updated with momentum 0.9, and the fitting steps are unchanged. Unlike the center loss of Wen et al. (2016), it acts on directions rather than Euclidean distances.

Instance-contrastive auxiliary. Two independently augmented views of each image are fitted. Each residual passes through a fixed random projection to 256 dimensions, an MLP with dimensions 256–256–128, and normalization. An InfoNCE loss (temperature 0.1, coefficient 0.1) treats the two views of an image as a positive pair and the other images in the batch as negatives, whatever their class. Classification and reconstruction use the first view; the second is fitted without differentiating through the gradients of its fitting steps.

Table A2. Settings that differ between configurations. The sampled fraction is the fraction of pixels sampled per fitting step.

Configuration	Sampled fraction	Feedback coefficient	Final pooling
Baseline	0.05	0.01	Mean over tokens
Composite	0.05	$0.001 \rightarrow 0.1$	Weighted per-layer means
Composite + denser fitting	0.10	$0.001 \rightarrow 0.1$	Weighted per-layer means
Bias encoder	0.05	0.01	Mean over tokens
Center auxiliary	0.05	0.01	Mean over tokens
Instance-contrastive auxiliary	0.05	0.01	Mean over tokens

A.3 Reporting conventions

The accuracy of a trained model is the Top-1 recorded for its checkpoint selected on validation data. Readouts are averaged over the three seeds of a configuration; summaries over configurations average these means, with the sample standard deviation (s.d.) across configurations.

Interventions use one seed of each configuration unless stated otherwise. Their effect is the change in Top-1 accuracy, in percentage points, from the same checkpoint evaluated on the unmodified input. Random draws are averaged within each checkpoint.

B Readouts of Class Structure

B.1 Held-out readouts

The linear and two-layer MLP readouts of Table 1 are trained on a fixed stratified 80% of the validation images (3,140) and tested on the other 20% (785). PCA is fitted on all validation images without labels, and features are standardized with the mean and standard deviation of the training part. Logistic regression selects C from $\{0.01, 0.1, 1, 10\}$ by five-fold cross-validation.

B.2 Readouts with training references

The 5-NN and centroid readouts of Table 1 use the 9,469 training images as references and all 3,925 validation images as queries, with Euclidean distance after standardization fitted on the training images. For the full residual, PCA is fitted without labels on the training and validation residuals.

At every dimension (Table B2), the readouts of the full residual rank the configurations at most moderately by classifier accuracy ($\rho \leq 0.43$).

Averaged over images, the centroid margin of the full residual (Section 3.2) ranks the configurations closely by classifier accuracy ($\rho = 0.94$ at 128 components), although its configuration means lie between -0.0006 and -0.0002 on a scale from -1 to 1. On average, an image thus lies about as close to another class centroid as to its own.

Table B1. The readouts of Table 1 with their standard deviations, mean ± s.d. over three seeds (%). The fitted parameters are the full residual reduced to 128 principal components. The 5-NN and centroid readouts use the training images as references (Appendix B.2), and the linear and MLP readouts the held-out split (Appendix B.1). The final-block output is averaged over tokens.

		Fitted parameters				Final block	
Configuration	Top-1	5-NN	Centroid	Linear	MLP	5-NN	Centroid
Baseline	60.7 ±0.8	19.9 ±0.2	41.0 ±0.2	39.8 ±0.1	37.7 ±1.6	55.8 ±1.3	58.1 ±0.9
Composite	62.3 ±0.2	20.7 ±0.5	41.9 ±0.2	40.4 ±0.3	41.2 ±1.4	58.9 ±0.3	59.6 ±0.5
Composite + denser fitting	62.6 ±0.7	21.5 ±0.5	41.6 ±0.2	40.6 ±0.8	40.3 ±0.4	60.4 ±0.8	60.8 ±1.2
Bias encoder	62.3 ±0.5	20.7 ±0.2	41.0 ±0.2	39.8 ±0.7	38.9 ±1.7	59.4 ±0.7	60.2 ±0.4
Center auxiliary	59.0 ±0.5	19.2 ±0.7	41.1 ±0.9	38.8 ±1.3	36.3 ±1.8	54.0 ±0.1	55.4 ±0.7
Instance-contrastive auxiliary	58.3 ±0.4	21.5 ±0.4	41.7 ±0.2	40.1 ±0.2	38.8 ±1.8	53.4 ±0.8	56.1 ±0.8

Table B2. Readouts with training references at several dimensions. Mean over the 18 trained models (six configurations, three seeds; %) and Spearman correlation ρ of the six configuration means with classifier Top-1.

	5-NN		Centroid	
Representation	Mean	ρ	Mean	ρ
Full residual, 64 PCs	30.9	0.43	40.1	0.09
128 PCs	20.6	0.14	41.4	0.09
256 PCs	13.0	−0.03	42.1	0.43
Input projection	28.2	0.09	26.9	−0.09
Final block	57.0	0.94	58.4	0.89

B.3 Classifiers trained afresh on frozen parameters

Here a classifier of the baseline architecture is trained afresh on the frozen fitted parameters of each configuration, taken from one seed, with the learned shift kept fixed. Each classifier is trained for 100 epochs with the cross-entropy loss only; we report the mean validation accuracy over the last five epochs, averaged over three classifier seeds. Unlike the jointly trained classifiers, all share one architecture, and the fitting does not adapt to them.

Table B3. Classifiers trained afresh on frozen fitted parameters (one seed per configuration). Top-1, mean ± s.d. over three classifier seeds (%).

Fitted parameters from	Top-1
Baseline	50.6 ±0.5
Composite	52.9 ±0.4
Composite + denser fitting	54.3 ±0.5
Bias encoder	53.5 ±0.1
Center auxiliary	51.9 ±0.4
Instance-contrastive auxiliary	51.7 ±0.3

Every classifier trained afresh (Table B3) is more accurate than the readouts of the fitted parameters in Section 4.2, and its accuracy tends to follow that of the jointly trained classifier of the same configuration (Spearman $\rho = 0.77$).

C Classifier-Input Interventions

C.1 Intervention definitions

A bias input is $b = \lambda(\Delta b + \beta_b)$, where β_b is the corresponding entry of the learned shift β (Equation 2). Each operation changes only the named inputs before the input projection. Replacing the biases by their layer means sets every bias of a SIREN layer to its mean over all images and neurons. Permuting

across images gives each image the bias vector of another image under a random permutation of the validation images; permuting within a class restricts the permutation to images of the same class (one draw). Removing the fitted residual keeps only $\lambda\beta_b$, removing the learned shift keeps only $\lambda\Delta b$, and single-layer zeroing sets the biases of one SIREN layer to zero. In each layer, the selected weight column is the column whose standard deviation is closest to that of the biases; it is replaced by its layer mean, set to zero, or permuted with the same permutation as the biases.

In Center auxiliary, the standard deviation of the selected column is within 16% of that of the biases in every layer and checkpoint; in the other four configurations it is 1.4–5.9 times smaller. For Center auxiliary, averaged over the three seeds, replacing the biases by their layer means costs 15.1 points and replacing the selected column 0.2 points.

C.2 Results by operation and configuration

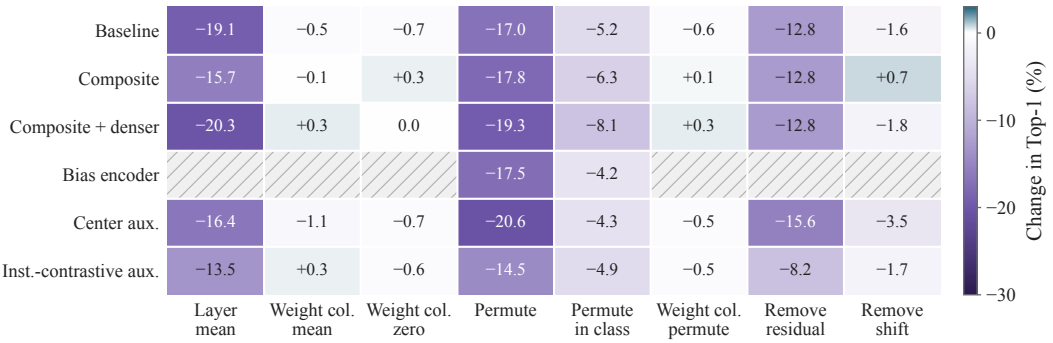

Figure C1. Intervention effects by configuration. Change in Top-1 (%) on one seed of each configuration, for the operations of Appendix C.1; columns act on the biases unless marked Weight col. (the selected weight column). Hatched: not applicable to Bias encoder.

Table C1. Zeroing about 1,024 values, by configuration (one seed per configuration). The columns set to zero all biases, one random weight column, the weights of four random tokens with their biases kept, or the same four tokens entirely (1,028 values). Random choices average five draws.

	Change in Top-1 (%)			
Configuration	All biases	Random column	Token weights	Whole tokens
Baseline	−18.3	0.0	−6.3	−0.2
Composite	−14.2	+0.1	−4.6	−0.4
Composite + denser fitting	−17.4	+0.6	−2.4	−0.4
Center auxiliary	−19.6	−0.2	−21.6	−0.3
Instance-contrastive auxiliary	−12.9	−0.1	−3.1	−0.1
Mean	−16.5	+0.1	−7.6	−0.3

Figure C1 shows each operation by configuration, and Table C1 the cost of zeroing about 1,024 values in each configuration.

Which class the permuted biases point to. After a permutation of the bias vectors across images, we compare each prediction with the class of the image that supplied the biases, over the images whose own class differs from it. On all 18 checkpoints, with five permutations each, the classifier predicts the supplying image's class for 11.9% of these images (10.1–15.1% across checkpoints), and the class of an unrelated image, drawn by a second, independent permutation, for 6.5% on average. The supplying image's class is the more frequent one on every checkpoint.

C.3 Other training seeds

The layer-mean replacement and permutation of the biases, the same operations on the selected weight column, and single-layer zeroing were repeated on the other two seeds of every configuration. On each of the fifteen checkpoints, replacing the biases costs at least 12.7 points and replacing the selected weight column at most 1.4 points (Table C2; Figure 4B). Averaged over the three seeds, zeroing the first-layer biases costs 8.1, 8.3, and 4.8 points for Baseline, Center auxiliary, and Instance-contrastive auxiliary, and permuting the biases of Bias encoder costs 18.5 points.

Table C2. Interventions on all three seeds. Change in Top-1 (%), mean ± s.d. over 15 checkpoints (five configurations, three seeds), or 18 for bias permutation and single-layer zeroing, which also apply to Bias encoder; permutations average five draws per checkpoint.

	All SIREN layers		One layer set to zero	
	Layer mean	Permute	Layer 1	Layer 4
Biases	−16.3 ±2.1	−17.9 ±1.7	−5.4 ±2.6	−0.4 ±0.9
Selected weight column	−0.3 ±0.5	−0.3 ±0.6	–	–

C.4 CIFAR-10

The layer-mean replacement of the biases and of the selected weight column was repeated on the CIFAR-10 models of Table 3 other than Bias encoder, that is, Baseline, Center auxiliary, Composite, and Composite + denser fitting at widths 128 and 256. The change in Top-1 is measured on the 10,000 test images. At width 256, replacing the biases costs 8.7 ± 2.0 points and replacing the selected weight column 0.1 ± 0.4 (mean ± s.d. over checkpoints). At width 128, the two costs are 14.6 ± 5.4 and 1.9 ± 2.2 points. Replacing the biases costs at least 5.2 points on every checkpoint.

D Color Statistics of the Biases

The 58 color and brightness statistics consist of RGB means and standard deviations for the full image and each cell of a 2×2 grid (30 features); value, saturation, and circular hue summaries (8); a 16-bin grayscale histogram; and grayscale mean, standard deviation, skewness, and kurtosis (4).

Let B be the matrix of bias vectors of the 3,925 validation images. For a design matrix X with standardized columns, we fit a ridge regression (penalty 1, with intercept) from X to B on these images and subtract the fitted values. X holds the 58 color statistics, 58 Gaussian random features, or the color statistics permuted across images. The variance control removes the four leading principal components of B (Table D1), which carry about as much variance as the color statistics explain.

Table D1. Removing the color-predictable part or leading principal components, by configuration (one seed per configuration).

	Change in Top-1 (%)		Variance removed (%)	
Configuration	Color	PCs	Color	PCs
Baseline	−16.0	−16.1	94.0	94.1
Composite	−12.7	−12.8	93.4	94.0
Composite + denser fitting	−15.7	−15.7	94.3	94.3
Center auxiliary	−16.8	−17.2	92.0	92.9
Instance-contrastive auxiliary	−12.5	−12.9	95.6	95.7

Random and permuted features remove 1.6% and 1.3% of the variance and cost 2.8 and 2.6 points on average. Because the regression has an intercept, every removal, including these controls, also sets each bias coordinate to zero mean over images.

On their own, the 58 standardized color statistics reach 40.8% Top-1 with a multinomial logistic regression ($C = 1$) trained on a stratified 70% of the validation images and tested on the other 1,178.

E Performance of Modified Configurations

Table 3 compares four configurations of Appendix A with the baseline. On CIFAR-10, the SIREN and classifier widths are 128 or 256, and the epochs and sampled fractions are those of Tables A1 and A2. Figure E1 shows differences from the baseline, and Table E1 gives per-seed accuracy.

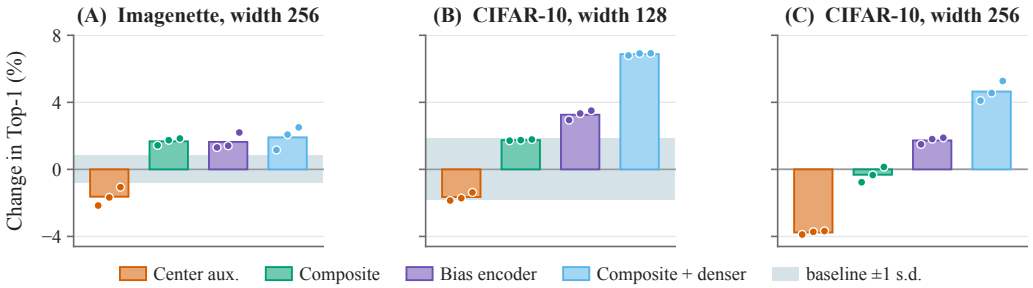

Figure E1. Difference in Top-1 from the baseline. Bars are means over three seeds (dots); the shaded band spans ±1 s.d. of the baseline seeds. Mean accuracies are in Table 3.

Table E1. Top-1 by seed (%), in the three settings of Table 3; network width in parentheses.

	Imagenette (256)			CIFAR-10 (128)			CIFAR-10 (256)		
Seed	1	2	3	1	2	3	1	2	3
Baseline	60.71	59.80	61.45	58.61	55.32	58.38	62.10	62.19	62.18
Center auxiliary	59.59	58.50	58.98	55.58	55.72	56.06	58.48	58.43	58.27
Instance-contrastive auxiliary	58.60	57.86	58.50						
Composite	62.50	62.09	62.39	59.22	59.18	59.15	62.30	61.39	61.82
Bias encoder	62.85	61.96	62.06	60.77	60.94	60.38	63.96	63.65	64.04
Composite + denser fitting	63.16	62.73	61.81	64.23	64.36	64.35	67.43	66.71	66.26

On Imagenette, the Center auxiliary loss is about 0.001 by the end of training, but the ten class mean directions are nearly parallel, with pairwise cosine similarities of 0.9996–1.0000 in all three seeds. The objective is thus met by moving all residuals toward one direction, not by separating the classes.