AVAILABILITY AND PREDICTIVITY: IMPROVING LOW-IPC CORESET TRAINING WITH COVARIANCE JITTER

Anonymous authors
Paper under double-blind review

ABSTRACT

Coreset selection aims to replace a large training set with a small subset while preserving downstream performance. We show that models trained on just a few images per class (IPC) rely on features along the data's high-variance directions, even when features along the low-variance directions are more predictive. This motivates our method, Coreset Covariance Jitter (CCJ), a simple, low-cost training augmentation applicable to any existing coreset selection method. CCJ adds noise to each selected image at every training step, scaled along each direction by how much images of the same class vary along it in the full training set. We further increase performance by adding a Quadratic Discriminant Analysis (QDA) at test time. On CIFAR-10, CIFAR-100 and ImageNet-1K, CCJ+QDA improves all evaluated coreset selection methods when running at 10 images per class.

1 INTRODUCTION

In recent years, the primary mechanism for improving the performance of deep learning models has been scaling up the amount of data and compute. However, this makes training more costly, and sharing datasets more difficult. One solution is coreset selection, which selects a small subset of the dataset, such that downstream performance matches that of a model trained on the full dataset. The difficulty is identifying which properties of the full dataset a small subset must preserve to achieve comparable downstream performance.

Existing methods match a variety of properties, such as geometry in feature space, training dynamics, difficulty scores and influence estimates (Guo et al., 2022). As the number of retained images grows, these methods perform well, but when the pruning ratio is high, accuracy drops dramatically. At low IPC (images per class), nonlinear models rely on high-variance features and neglect the low-variance features that are important for generalisation (Section 3). To counter this, we introduce Coreset Covariance Jitter (CCJ), an augmentation that can be applied alongside the coreset to significantly improve accuracy at negligible compute and storage cost (around 0.34 extra images per class for CIFAR-100 and 0.01 for ImageNet-1k).

CCJ perturbs each coreset image while training the downstream student model. The strength of the perturbation in each dimension is based on the data's variance in that direction, so features in high-variance subspaces are perturbed more than the features in low-variance subspaces. This makes high-variance features less reliable during training, so the model shifts toward low-variance features, which it would otherwise learn only with more images. As CCJ steers the network away from the high-variance subspace, we recover the class information that subspace has with Quadratic Discriminant Analysis (QDA) and combine it with the trained network at test time. Our method is designed to be the most effective in the challenging low-IPC setting.

Contributions.

1. We give a theoretical account of why models trained on few images per class prefer high-variance features. We demonstrate that nonlinear models focus on a few dominant directions in the pixel space when trained with few images per class, and as we increase the IPC, the model starts to learn patterns in the remaining directions.

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.

2. We introduce Coreset Covariance Jitter (CCJ), a method to encourage networks to learn features present in the low-variance directions. We show it improves all 16 selectors on CIFAR-100 and all 11 on ImageNet-1K at IPC-10.
3. The introduction of Quadratic Discriminant Analysis (QDA) as an explicit classifier and combining it as a product of experts, which adds a further $+3.53$ to $+5.02$ points to CCJ for every selector on CIFAR-100 at IPC-10.

2 Related work

Most coreset selection methods rank the training images by a scalar score and keep the top-n of the ranking. Geometric methods aim to preserve the structure of data in the representation space through herding (Welling, 2009) or greedy k-centre (Sener & Savarese, 2018). A second line of work estimates sample importance from the training dynamics or difficulty, using a proxy such as forgetting events (Toneva et al., 2019), the gradient and error norms GraNd and EL2N (Paul et al., 2021), the area under the margin (Pleiss et al., 2020), predictive entropy from a proxy model (Coleman et al., 2020), dynamic uncertainty (He et al., 2024), temporal dual-depth scoring (Zhang et al., 2024) and the difficulty-uncertainty product of DUAL (Cho et al., 2025). Influence-based pruning instead estimates how removing a set of training samples would change generalisation, and removes the largest set whose estimated effect stays within a bound (Yang et al., 2023). Rather than simply retaining the highest-scoring samples, several methods explicitly incorporate diversity or coverage into selection. CCS stratifies the importance-score distribution (Zheng et al., 2023), D^2 propagates difficulty across a k-NN graph (Maharana et al., 2024), InfoMax penalises redundancy between the selected points (Tan et al., 2025). GLOBE instead aligns the selection with the training trajectory through structured sparse optimisation (Liu et al., 2026). Finally, uncertainty-based methods use uncertainty scores to guide selection, such as CAL (Margatina et al., 2021), DeepFool (Ducoffe & Precioso, 2018), and classical uncertainty rules (Settles, 2009). Our analysis of low IPC model behaviour broadly applies to all of these approaches. Based on this analysis, we propose CCJ, which composes naturally with any of these selectors. Dataset distillation (Wang et al., 2018) instead synthesises the stored images by optimising them against the full training set; CCJ and QDA are complementary to it, and improve distilled sets as well (Appendix A.8).

Our method adds noise to different dimensions of the input image based on their variance. Prior work has long used perturbed inputs as a form of data augmentation. Vicinal risk minimisation replaces each training point with a distribution around it (Chapelle et al., 2001), tangent propagation penalises the derivative along a known invariance (Simard et al., 1992), and additive input noise is equivalent to Tikhonov regularisation (Bishop, 1995). Crop-and-flip pipelines (Krizhevsky et al., 2012) and mixup (Zhang et al., 2018) choose their perturbations without reference to class statistics, and mixup interpolates between classes rather than inside one. ISDA (Wang et al., 2019) estimates the class-conditional covariances online and uses them to augment deep features implicitly, rather than perturbing the inputs.

3 Theory

Prior work has examined how dataset size (Bordelon et al., 2020; Canatar et al., 2021), eigenvalues (Saxe et al., 2014; Simon et al., 2023) and frequency (Rahaman et al., 2019) affect the order in which features are learned, but not which of two predictive features a model relies on when they disagree. Studies of simplicity bias and feature availability (Shah et al., 2020; Hermann et al., 2024) show that models prefer the more available feature even when it is less predictive, but not how much more predictive a less available feature must be before the model switches to it, or how this threshold changes with IPC. We derive this margin in two claims, by separating a feature's availability (set by the variance of the data and by the architecture) from its predictivity.

Claim 1 (Dependence of Availability on Variance) *Features encoded along directions where the data varies most are more available to the student model than features encoded along low-variance directions. Therefore, a low-variance feature must be more predictive than a high-variance feature (enough to compensate for its lower availability) before the learner exploits it.*

Claim 2 (Relationship between Predictive Advantage and Images Per Class (IPC)) *Increasing the number of images per class increases how much of each feature the model learns, up to a saturation point, and less-available low-variance features gain the most. Therefore, as IPC increases, the predictive advantage a low-variance feature needs before the learner exploits it falls.*

We give a theoretical account of both claims, with proofs in Appendix E, and then test them empirically in a linearised network (Appendix C.1) and a nonlinear ConvNet (Section 3.4).

3.1 Planted Task

Testing these claims requires separating feature availability from predictive power. Natural images do not allow this, since we cannot choose where the class information lies or how strong it is. We therefore construct an artificial task that plants class information, of controlled strength, in both a high-variance and a low-variance subspace. We then ask which the model prefers. Each training input combines a class label $y \in \{1, \dots, C\}$ (drawn uniformly) with a background image $x_0 \in \mathbb{R}^D$ drawn independently of y. Because the labels and backgrounds are randomly assigned, the class can only be predicted from the features we plant into it. Let μ and Σ be the mean and covariance of the backgrounds, with eigendecomposition $\Sigma = P\Lambda P^\top$, where the columns of $P = [u_1, \dots, u_D]$ are the eigenvectors of Σ and $\Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_D)$ are their descending eigenvalues. We choose two K-dimensional subspaces: a high-variance subspace (P_{hi}, Λ_{hi}) and a low-variance subspace (P_{lo}, Λ_{lo}).

We construct the planted information so we can easily disentangle its predictivity from its availability. We give each class (c) a unique signal in each subspace by choosing random unit vectors ($\nu^c_{hi}, \nu^c_{lo} \in \mathbb{R}^K$), and scaling by their respective standard deviations ($\Lambda^{1/2}$) to ensure they are equally informative about the label (i.e. equally predictive)

$$d^{hi}_c = l P_{hi} \Lambda^{1/2}_{hi} \nu^c_{hi}, \qquad d^{lo}_c = t l P_{lo} \Lambda^{1/2}_{lo} \nu^c_{lo}. \tag{1}$$

Here $l > 0$ controls the absolute scale of both signals and $t > 0$ controls the relative strength of the low-variance feature compared to the high-variance feature. Each planted training sample is $x_i = x_{0,i} + d^{hi}_{y_i} + d^{lo}_{y_i}$, where $x_{0,i}$ is the i-th background. We use n samples per class (the IPC), $N = Cn$ in total.

The training dynamics of a full network are too complex to analyse directly, so following Jacot et al. (2018) and Lee et al. (2019), we use a network linearised in its parameters around its initialisation. It is still a nonlinear function of the image, but its tangent kernel $\Theta(x, x')$ (which measures how much a gradient step on image x moves the output on a different image x') stays fixed during training (Appendix E.1). This lets us determine how quickly each planted feature is learned.

The feature a model utilises can change during training, so we track its behaviour. We train the linearised network using cross-entropy via unregularised gradient flow with learning rate η (Appendix E.1), and write $f_\tau(x) \in \mathbb{R}^C$ for its output at training time $\tau \geq 0$.

To see which feature the model relies on at a given time, we put them in conflict. A *conflict input* $x_{ab} = x_0 + d^{hi}_a + d^{lo}_b$ plants the high-variance feature of class a and the low-variance feature of a different class b into a new background x_0. The model's preference is then measured as its margin (i.e. the difference between the logits) of the two classes, $m_{ab}(\tau) = (e_a - e_b)^\top f_\tau(x_{ab})$, where e_c is the class-c one-hot vector. This margin is positive when the model follows the high-variance feature and negative when it follows the low-variance one. Our analysis makes the following assumption.

Assumption 1 (Planted regime) *(a) The backgrounds are Gaussian, $x_0 \sim \mathcal{N}(\mu, \Sigma)$. (b) The tangent kernel is class-diagonal, i.e. a gradient step on one class's logit at x moves only that class's logit at x', by an amount $\Theta(x, x')$ which is the same for every class. (c) The tangent kernel is twice differentiable with integrable derivatives. (d) At initialisation, the logits are centred, and the softmax output is uniform over classes. (e) The planted features' strength l is small relative to background variation.*

We discuss when these assumptions hold in Appendix B.

3.2 Availability and Predictivity in the Planted Task

In order to prove claim 1, we must define two properties of a feature: how useful it is for predicting the label (its predictivity), and how easily the model learns it (its availability). We now formalise both for the planted task. To measure predictivity, we ask how much weight an ideal classifier would give each feature. For Gaussian backgrounds, the Bayes-optimal classifier would weight each feature according to its whitened coordinates (Lemma 1 in Appendix E). We can then measure the predictivity of a feature by its whitened strength

$$\rho_{hi} := \|\Sigma^{-1/2} d_c^{hi}\| = l, \qquad \rho_{lo} := \|\Sigma^{-1/2} d_c^{lo}\| = tl. \tag{2}$$

Hermann et al. (2024) defines the availability of a feature as how easily it can be learned by a given model. In our setting, we formalise this by measuring how fast a feature is learned at the start of training. Lemma 2 in Appendix E proves that the model's margin for the true class over the other classes grows at a rate proportional to two terms: (1) The feature's predictivity (ρ_q^2), (2) ζ_q, which depends only on the model and the backgrounds, not on the feature itself. We call this second term the subspace's availability.

The availability $\zeta(v)$ of a particular direction v measures how quickly the model initially learns a feature planted along v. It is set by the tangent kernel: how much a gradient step on one background moves the output on another, in proportion to how far both lie along v.

To formalise this, we first specify $\varphi_v(x)$ as the whitened coordinate of x along direction v (i.e. $\varphi_v(x) = \sum_j v_j \varphi_j(x)$) where $\varphi_j(x)$ is the whitened coordinate along eigendirection u_j. The availability is then

$$\zeta(v) := \mathbb{E}_{x,x'}\left[\Theta(x,x')\,\varphi_v(x)\,\varphi_v(x')\right], \tag{3}$$

where x, x' are independent backgrounds. Because each class direction ν^c is random, a feature is, on average, spread evenly across the eigendirections of its subspace, so the availability of the subspace is the average availability of its eigendirections,

$$\zeta_q = \frac{1}{K}\sum_{j\in q} \zeta(u_j). \tag{4}$$

3.3 Preferences of the Model while Training

Now that we have defined the availability and predictivity individually, we can derive how they combine to determine which features the model prefers during training. Two quantities define the preference. The first is the threshold $t_\star(\tau)$. This is the strength t for which the model is indifferent between the two features on a conflicting input, at time τ. The second, $L_q(\tau)$ is how much of the feature in subspace $q \in \{hi, lo\}$ the model has learned by time τ, as a fraction of the Bayes-optimal weight ρ_q. At initialisation $L_q(0) = 0$ and when the model weights the feature exactly as the Bayes predictor does $L_q = 1$. Formally, it measures how strongly the model's margin for a class c over a competitor $c' \neq c$ follows the background's whitened coordinate along the class direction ν_q^c, and it is given by,

$$L_q(\tau) := \frac{1}{\rho_q}\,\mathbb{E}_{x_0}\left[(e_c - e_{c'})^\top f_\tau(x_0)\,\varphi_{\nu_q^c}(x_0)\right]. \tag{5}$$

To define $L_q(\tau)$ beyond the start of training, we need one further assumption. This assumption ensures that features along each whitened direction are learned independently, at a rate set by their availability.

Assumption 2 (Kernel diagonal in the whitened basis) *The whitened coordinates are eigenfunctions of the kernel, i.e. for every direction j,*

$$\mathbb{E}_{x'}\left[\Theta(x,x')\varphi_j(x')\right] = \zeta_j \varphi_j(x), \tag{6}$$

where x' is an independent background and ζ_j is the availability in eigendirection u_j (Eq. 3).

We now have everything we need to derive $L_q(\tau)$ and the relationship between $t_\star$ and $L_q(\tau)$. Intuitively, on a conflict input each feature pulls the margin towards its own class with strength $\rho_q^2 L_q(\tau)$ (its squared predictivity times the fraction of it which has been learned). The threshold is where the two pulls balance. The full proofs are in Appendix E but in summary:

Proposition 1 (preference along training) *Under Assumptions 1 and 2, for every training step $\tau \geq 0$ the preference changes sign at*

$$t_\star(\tau)^2 \approx \frac{L_{hi}(\tau)}{L_{lo}(\tau)}. \tag{7}$$

In particular, $t_\star(\tau)^2 \to \zeta_{hi}/\zeta_{lo}$ as $\tau \to 0$.

With infinite data, we can go further and solve for $L_q(\tau)$. Each whitened direction is then learned independently, at its own rate $\eta\zeta_j/C$, so the more available high-variance feature is recovered first.

Proposition 2 (recovery in the population limit) *Under Assumptions 1 and 2, in the population limit $N \to \infty$*

$$L_q(\tau) \approx \frac{1}{K}\sum_{j\in q}\left(1 - e^{-\eta\zeta_j\tau/C}\right). \tag{8}$$

If every high-variance direction is more available than every low-variance one ($\zeta_j > \zeta_k$ for all $j \in hi$, $k \in lo$), then $1 \leq t_\star(\tau)^2 \leq \zeta_{hi}/\zeta_{lo}$ for every $\tau > 0$, and $t_\star(\tau)$ decreases strictly to 1 as $\tau \to \infty$.

The threshold therefore starts at the availability ratio $\sqrt{\zeta_{hi}/\zeta_{lo}}$ and falls towards 1 during training, as the high-variance feature saturates and the low-variance one catches up. Note that it never drops below 1. At every point in training, a low-variance feature must be more predictive than its high-variance competitor before the model follows it.

This relies on the high-variance directions being the more available ones, so to prove Claim 1 we must connect availability to the eigenvalues. Because the backgrounds are Gaussian (Assumption 1), Stein's identity (Stein, 1981) gives this connection. Each eigendirection splits into a data term (its variance λ_j) and an architecture term (the model's input sensitivity g_j along that direction).

Proposition 3 (availability factorises into data and architecture) *Under Assumption 1, the availability of each eigendirection, $\zeta_j := \zeta(u_j)$, factorises as*

$$\zeta_j = \lambda_j\, g_j, \qquad g_j := u_j^\top G\, u_j, \qquad G := \mathbb{E}_{x,x'}\left[\nabla_x \nabla_{x'} \Theta(x, x')\right]. \tag{9}$$

Hence $\zeta_q = \frac{1}{K}\sum_{j\in q}\lambda_j g_j$.

Proof of Claim 1. By Propositions 1 and 3, at any dataset size the threshold early in training is $t_\star^2 = \zeta_{hi}/\zeta_{lo} = \sum_{j\in hi}\lambda_j g_j \big/ \sum_{j\in lo}\lambda_j g_j$. Assuming the architecture does not reverse the variance ordering (i.e. $\lambda_j/\lambda_k > g_k/g_j$ for every $j \in hi$ and $k \in lo$), then every high-variance direction is more available than every low-variance one. A low-variance feature must then be more predictive by the factor $\sqrt{\zeta_{hi}/\zeta_{lo}}$ before the model follows it. This is the hypothesis of Proposition 2, therefore in the population limit a low-variance feature must be more predictive at every training step τ, although the factor it needs does fall towards 1.

Proof of Claim 2. By Eq. 7, $t_\star$ falls with IPC when L_{hi}/L_{lo} does. We theoretically prove the fall of L_{hi}/L_{lo} with IPC when using a squared loss in the appendix (Proposition 6). In the case of a cross entropy loss, we will instead measure it empirically in the following section.

3.4 Testing the Results in a Nonlinear ConvNet

We test Claims 1 and 2 empirically in a nonlinear ConvNet. We use the Planted Task setting (Section 3.1) with CIFAR-10 training images as the background, discarding the original labels and sampling new ones independently. We take the subspace spanned by the top 20 eigenvectors as the high-variance subspace, and the next 20 as the low-variance subspace. Appendix C repeats the experiments with alternative low variance subspaces.

We first measure the availability of every eigendirection of the backgrounds (Eq. 3) for the ConvNet at initialisation, averaging over the $50,000$ CIFAR-10 training images. Figure 1(a) plots this against variance for the leading 366 directions (weaker directions are hard to distinguish from noise), and shows that availability and variance are positively correlated (Spearman correlation 0.78). The top 20 directions are 9.1 times as available as the next 20. The kernel sensitivity g_j varies by more than an order of magnitude between directions but barely correlates with their variance (Spearman

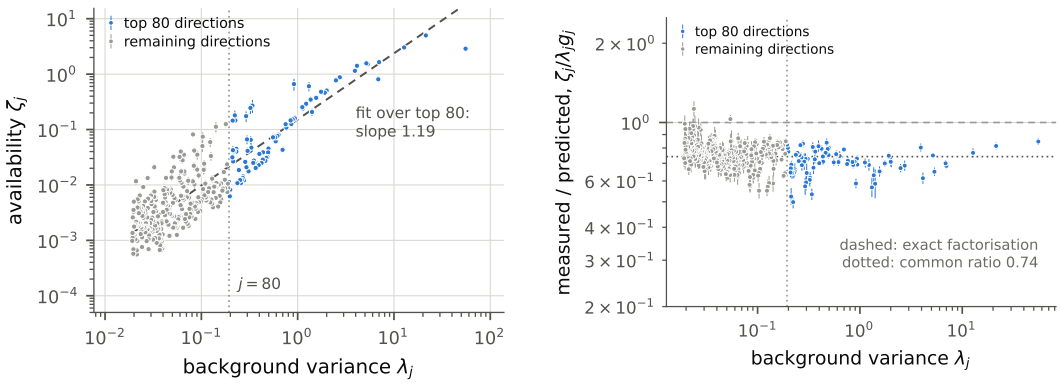


Figure 1: **Availability of each eigendirection in the ConvNet at initialisation.** (a) Measured. (b) Agreement between measured and predicted by Proposition 3.

correlation 0.17), so it scatters the availability of individual directions without reversing the ordering the variance sets. The ordering therefore holds, so the condition for Claim 1 is satisfied empirically.

To verify the effects of IPC on the subspace preference of the model, we trained the ConvNet on the planted task at different IPCs, ranging from five to a thousand images per class. We then varied the relative predictive strengths of the features planted in the low-variance subspace (via t), in order to empirically determine the indifference point $t_\star$. Figure 2 shows the resulting measured thresholds. At ten images per class, as the low-variance-planted features are made more predictive, the preference for the high-variance feature weakens and eventually switches at $t_\star = 2.28$. As IPC increases, the crossing point falls, levelling off beyond about 100 images per class. To compare this with the theory, we measure the recovered fractions $L_{hi}(t)$ and $L_{lo}(t)$ of the same trained networks by Eq. 5, taking the expectation over the $10,000$ CIFAR-10 test images as background x_0. We then use Eq. 7 to calculate the predicted threshold and show it in Figure 2. The measured and predicted indifference points agree to within about 0.1 at every IPC.

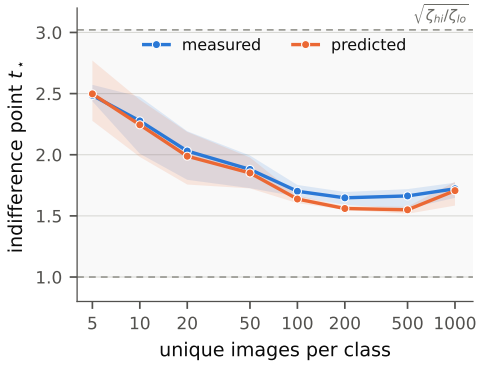


Figure 2: **The threshold $t_\star$ against IPC.**

4 Method

4.1 Coreset Covariance Jitter

Claims 1 and 2 imply that at low IPC the model follows the high-variance features unless a low-variance feature is much more predictive. To counter this, we reduce reliance on features in the high variance subspace by limiting their availability and predictivity while preserving class information. We therefore propose Coreset Covariance Jitter (CCJ), given a stored image x of class c, we add a fresh perturbation at every training step. The perturbation is drawn along the eigendirections of the within-class covariance $\Sigma_c \in \mathbb{R}^{D\times D}$, with a strength set by each direction's eigenvalue,

$$\tilde{x} \;=\; \mathrm{clip}_{[0,1]}\big(x+\varepsilon\big), \qquad \varepsilon \sim \mathcal{N}\big(0,\; s^2\Sigma_c\big). \tag{10}$$

We use the within-class rather than the total covariance because the total covariance also includes the between-class directions, and jittering along these would perturb the class signal itself.

Including the full Covariance Matrix alongside the coreset costs much more than storing just the images, so Eq. 10 must be approximated. Instead of the eigenbasis of the training set's covariance, we use a fixed orthonormal basis that approximates the eigenbasis of natural-image covariance, and so does not need to be stored. It combines two orthogonal matrices, one for chromatic correlation and one for spatial correlation. Following Ohta et al. (1980), the chromatic factor is the 3×3 matrix whose rows are $(1, 1, 1), (1, 0, -1), (1, -2, 1)$ row-normalised. In colour terms, these are luminance, red-blue and magenta-green, an approximate eigenbasis of the covariance of colour channels in natural images. The spatial factor is the orthonormal DCT-II (Ahmed et al., 1974), whose basis vectors are cosines of increasing frequency and approximate the eigenbasis of the spatial covariance of natural images. Taking the Kronecker product of the chromatic factor with the DCT-II along each spatial axis gives an orthonormal basis $B \in \mathbb{R}^{D \times D}$, whose rows we call atoms. Each atom combines one colour component with one pair of spatial frequencies.

We then replace Σ_c in Eq. 10 by a matrix that is diagonal in this basis:

$$\tilde{x} \;=\; \mathrm{clip}_{[0,1]}\big(x + \varepsilon\big), \qquad \varepsilon \sim \mathcal{N}\big(0,\; s^2 B^\top \mathrm{diag}_a\big(w_a^2\, \sigma_a^2\big) B\big), \tag{11}$$

where s controls the overall jitter strength, σ_a^2 is the within-class variance of atom a (pooled over classes), and w_a is a per dimension scaling factor that is close to one for the most variable atoms and falls smoothly towards zero for the rest. More formally,

$$w_a \;=\; \Big[1 + \big(\sigma_\star^2 / \sigma_a^2\big)^\beta\Big]^{-1}, \tag{12}$$

where $\sigma_\star^2$ is the variance of the $K_\star$-th most variable atom, and β sets how sharply the weight falls.

This approximation is also far cheaper to store, which is vital in coreset selection. Storing each Σ_c would cost $D(D+1)/2$ floats per class, the equivalent of $2(D+1)$ *uint8* images, or over $6{,}000$ images per class at 32px. In contrast, the basis B is fixed and does not depend on the dataset, and the variances are pooled over classes, so in addition to the coreset images we only need to store one variance for each atom. That amounts to D floats in total, which is the equivalent of 4 *uint8* images, regardless of the number of classes.

We apply Eq. 11 to the stored image after the usual crop, flip and colour augmentation. We set $K_\star = 80$, since in the ConvNet experiment the top 80 eigendirections of the backgrounds contained 92% of the total availability (Figure 1(a)). CCJ is relatively insensitive to this choice: at IPC $= 10$, accuracy stays within 0.43 points for any $K_\star$ between 20 and 160 (Appendix A.7.2).

By Claim 1, weakening the high-variance features in this way should shift the model towards the low-variance ones. By Claim 2, the benefit of this will be largest at low IPC (where the preference imbalance is strongest) and shrink as the coreset grows. We test both predictions in the planted task (Appendix D) and on real coresets in Section 5.

4.2 Quadratic Discriminant Analysis (QDA)

CCJ encourages the model to learn more difficult, predictive features when they exist in the low-variance subspace. However, this can adversely affect the learning of coarse class-level cues (such as each class's typical colour and overall layout) which are contained in the high-variance subspace. A complex network does not need to learn these simple cues. We therefore introduce a second, simple expert, a quadratic discriminant that models each class as a Gaussian in the high-variance subspace. Combining a CCJ-trained student network with the QDA model gives an ensemble of two experts, each tuned to extract information effectively from a different subspace.

To build this expert, we partially reuse the basis of the jitter (B), keeping only its r most variable atoms to represent the high-variance subspace ($A \in \mathbb{R}^{r \times D}$). The QDA model then scores each class by the log-likelihood ($\ell_c(x)$) of the image's projection (Ax) under that class's Gaussian, $\mathcal{N}(A\mu_c, S_c^{\mathrm{dct}})$, where μ_c is the mean of the class and $S_c^{\mathrm{dct}} = A\,\Sigma_c A^\top$ is the covariance within this subspace. Specifically

$$\ell_c(x) \;\propto\; -\tfrac{1}{2}\,\big(A(x-\mu_c)\big)^\top \big(S_c^{\mathrm{dct}}\big)^{-1}\,\big(A(x-\mu_c)\big) \;-\; \tfrac{1}{2}\log\det S_c^{\mathrm{dct}}. \tag{13}$$

Table 1: **CCJ across different coreset selection methods**

	IPC-10 (98% pruned)				IPC-50 (90% pruned)			
selector	std	+CCJ	std Q	+CCJ Q	std	+CCJ	std Q	+CCJ Q
full dataset	77.17±0.59							
random	15.01±0.32	19.94±0.48	19.53±0.39	**24.09**±0.53	45.35±1.16	48.06±0.19	46.47±0.91	**49.99**±0.26
forgetting	11.62±0.41	12.98±0.24	15.84±0.34	**17.44**±0.39	33.94±0.35	37.64±0.34	37.31±0.30	**41.58**±0.27
AUM	20.39±0.35	25.62±0.23	25.03±0.40	**30.52**±0.08	53.27±0.55	52.96±0.28	51.76±0.47	**53.35**±0.66
Dyn-Unc	11.09±0.11	14.71±0.33	15.03±0.26	**19.24**±0.33	36.68±0.63	38.99±0.15	39.81±0.42	**42.96**±0.17
TDDS	15.55±0.52	21.28±0.10	20.15±0.49	**26.30**±0.12	46.02±0.98	47.23±0.38	47.40±0.95	**49.20**±0.13
DUAL	22.35±0.33	28.73±0.70	25.90±0.56	**32.32**±0.66	**54.44**±0.41	54.14±0.26	52.99±0.32	54.38±0.30
EL2N	8.15±0.25	11.44±0.52	11.30±0.35	**15.28**±0.36	24.57±0.24	27.97±0.06	28.58±0.38	**32.35**±0.18
GraNd	9.54±0.26	12.80±0.33	12.80±0.31	**16.33**±0.12	35.78±0.38	38.58±1.54	38.78±0.50	**42.20**±1.44
entropy	10.40±0.09	13.49±0.23	13.51±0.32	**17.04**±0.38	34.69±0.44	39.90±0.56	37.61±0.23	**42.79**±0.33
least-confidence	11.28±0.22	13.39±0.33	14.89±0.24	**17.35**±0.31	39.97±1.17	41.72±0.59	42.12±0.95	**44.67**±0.47
margin	10.61±0.31	14.08±0.19	14.17±0.32	**18.01**±0.18	41.68±0.22	43.74±0.23	43.64±0.25	**46.50**±0.39
CAL	18.77±0.28	23.96±0.45	23.09±0.38	**28.70**±0.55	43.26±0.12	46.04±0.43	43.87±0.11	**47.56**±0.35
DeepFool	11.12±0.19	14.72±0.23	14.74±0.07	**19.10**±0.29	39.63±0.80	43.91±0.65	42.16±0.61	**46.71**±0.51
CCS	22.63±0.19	28.47±0.32	26.37±0.09	**32.26**±0.55	**51.83**±0.28	51.11±0.14	50.66±0.33	51.15±0.14
D^2	21.37±0.15	27.14±0.60	25.32±0.19	**31.68**±0.49	52.75±0.49	52.54±0.21	52.12±0.23	**53.29**±0.10
Herding	16.40±0.43	22.19±0.48	21.12±0.42	**26.70**±0.47	46.91±0.39	48.84±0.62	47.60±0.46	**50.51**±0.62

The class means $A\mu_c$ and covariances S_c^{dct} are estimated on the full training set and stored alongside the coreset. They cannot be estimated from the coreset itself: at IPC-10 each class's $r \times r$ covariance would rest on only 10 images. This requires $C\big(r + r(r{+}1)/2\big)$ floats, the equivalent of about 0.3 *uint8* images per class at 32px. We set $r = 20$. At test time, we combine the student and the QDA as a product of experts, summing their log-probabilities,

$$\hat{y}(x) \;=\; \arg\max_{c} \;\Big[\log \mathrm{softmax}\big(f(x)\big)_c \;+\; \log \mathrm{softmax}\big(\ell(x)/T_{\mathrm{cal}}\big)_c\Big], \tag{14}$$

where f is the trained network and T_{cal} is a single temperature per coreset, fit by minimising the QDA's negative log-likelihood on the coreset itself. Appendix D.3 additionally analyses the behaviour of the QDA in the planted task of Section 3.4, finding that it strongly prefers the high-variance features.

5 Experiments and Results

We apply our method to the coresets chosen by existing selection methods on CIFAR-100 and ImageNet-1K. In both cases the student is a ResNet-18 trained from scratch under a cross-entropy loss, for 200 epochs on CIFAR-100 and 300 on ImageNet-1K. On CIFAR-100 most selectors already cover every class, so we match only the total image count. On ImageNet-1K we enforce class-balanced selection for every method.

For CIFAR-100, IPC-10 corresponds to a pruning ratio of 98% and IPC-50 corresponds to 90% pruning; in ImageNet-1K, IPC-10 is 99.2% pruning, and IPC-50 is 96.1% pruning. Like the selectors they build on, CCJ and QDA assume the full labelled training set is available when the coreset is built. This is standard: selectors such as forgetting, AUM, TDDS and DUAL train networks on the full training set to score each image, whereas CCJ and QDA only compute per-atom variances and low-rank class statistics from it. On CIFAR-100 these cost the equivalent of about 0.34 images per class, so a coreset at IPC-10 with CCJ and QDA stores the equivalent of 10.34 images per class. On ImageNet-1K, where each image is much larger, the cost is about 0.01 images per class. Computing them is a one-off cost of under a second on CIFAR-100 and about 45 minutes on ImageNet-1K. CCJ adds only 2–4% to the time of each training epoch, and QDA 1–3% to inference (Appendix H). All tables compare methods at the same number of images.

5.1 Results in Low IPC in CIFAR-100 and ImageNet-1K

At IPC $= 10$, CCJ with QDA improves every selector on CIFAR-100 (Table 1) and ImageNet-1K (Table 2). CCJ gains most on strong coresets; its gain depends on the amount of information present in the low-variance subspace and how well the model already learns it. QDA gains most on weak coresets, because it changes the outputs of those models more than those already trained on strong coresets. At IPC $= 50$, the weak selectors still improve, but the strongest gain little or lose up to 1.28 points. On ImageNet-1K these losses come from QDA, which lowers accuracy for 7 of the 11

Table 2: **Evaluation in ImageNet-1K.**

	IPC-10 (99.2% pruned)				IPC-50 (96.1% pruned)			
selector	std	+CCJ	std Q	+CCJ Q	std	+CCJ	std Q	+CCJ Q
full dataset†	69.76							
random	10.46±0.08	14.83±0.08	11.88±0.08	**16.24**±0.05	38.41±0.33	**39.76**±0.25	38.08±0.39	39.53±0.36
forgetting	12.65±0.08	16.96±0.13	13.94±0.13	**18.13**±0.13	46.34±0.31	**47.00**±0.30	44.96±0.52	45.83±0.52
AUM	19.01±0.14	**22.26**±0.27	18.98±0.14	22.16±0.19	44.68±0.14	**45.28**±0.04	42.66±0.10	43.45±0.13
Dyn-Unc	13.83±0.35	18.60±0.15	15.03±0.33	**19.63**±0.24	45.51±0.17	**46.09**±0.07	44.30±0.15	45.02±0.19
TDDS	12.76±0.16	17.02±0.17	14.05±0.16	**18.21**±0.22	38.82±0.15	**40.41**±0.23	38.19±0.15	39.76±0.34
DUAL	6.93±0.09	10.77±0.26	8.33±0.11	**12.38**±0.27	40.97±0.16	41.93±0.22	40.89±0.22	**42.05**±0.24
EL2N	4.55±0.08	6.32±0.18	5.68±0.15	**7.67**±0.23	19.56±0.20	21.19±0.29	20.97±0.22	**22.83**±0.24
entropy	5.58±0.23	8.94±0.20	6.79±0.27	**10.26**±0.12	27.98±0.23	30.14±0.08	28.75±0.29	**30.95**±0.08
least-confidence	3.31±0.13	5.72±0.14	4.18±0.12	**6.69**±0.12	26.18±0.43	27.93±0.24	27.11±0.40	**28.83**±0.23
margin	6.50±0.04	9.25±0.32	7.74±0.07	**10.66**±0.35	35.48±0.30	37.18±0.29	35.72±0.20	**37.61**±0.30
CAL	18.57±0.11	**21.28**±0.25	18.39±0.09	21.13±0.24	40.68±0.10	**41.17**±0.16	38.98±0.04	39.40±0.20

Table 3: **CCJ across student architectures.**

	IPC-10				IPC-50			
student	std	+CCJ	std Q	+CCJ Q	std	+CCJ	std Q	+CCJ Q
ResNet-18	22.68±0.86	29.71±0.27	25.97±0.83	**33.13**±0.10	53.92±0.20	**54.84**±0.18	52.95±0.22	54.41±0.30
ConvNet-3	21.50±0.36	24.03±0.14	25.83±0.22	**29.30**±0.03	40.17±0.18	44.03±0.32	41.37±0.35	**45.20**±0.21
ResNet-50	22.85±0.31	26.74±0.86	25.93±0.41	**30.64**±1.00	48.37±1.34	51.19±0.83	48.35±1.15	**51.98**±0.64
VGG-11 (BN)	23.14±0.67	22.27±0.56	26.39±0.40	**28.19**±0.36	48.18±0.87	48.89±0.45	47.86±0.89	**49.18**±0.54
DenseNet-121	24.51±0.47	31.06±0.09	27.88±0.42	**34.11**±0.12	54.82±0.44	**54.96**±0.31	53.62±0.29	54.93±0.21
ViT-Tiny/4	14.42±0.50	18.77±0.41	20.21±0.30	**25.04**±0.22	28.93±0.47	38.17±0.07	32.72±0.23	**40.44**±0.12

selectors at this budget, while CCJ alone improves all of them. A more detailed analysis is provided in Appendix A.1. Under strong augmentation (RandAugment + CutMix + Cutout; Cubuk et al., 2020; Yun et al., 2019; DeVries & Taylor, 2017), our method still improves every selector by +3.76 to +9.06 points at IPC $= 10$ on CIFAR-100 (Appendix A.6.1).

5.2 CCJ in a Label-Free Setting

Following ELFS (Zheng et al., 2025), we generate pseudo-labels by k-means clustering and compute the within-class statistics over clusters instead of classes, so the σ_a^2 of Eq. 11 become within-cluster variances pooled over clusters. Without labels, CCJ still improves both random and ELFS coresets by $+0.80$ to $+9.20$ points on CIFAR-10 and CIFAR-100 at 1, 10 and 50 IPC. To match ELFS, this experiment uses a slightly different protocol, with a ConvNet-3 student (Appendix A.4).

5.3 Effect of CCJ on Different Student Architectures

CCJ and QDA together improve accuracy on all six student architectures (Table 3), by $+5.05$ to $+10.62$ points at IPC-10 and by $+0.11$ to $+11.51$ at IPC-50. CCJ also helps without QDA: it improves all six students at both IPC levels (Table 10). In the planted task (Section 3.1), every student prefers the high-variance subspace at low IPC, and CCJ lowers the preference for every student. The gains vary between students for the same reasons they vary between selectors (Section 5.1): CCJ helps most for students that otherwise neglect the low-variance subspace, and QDA helps the weakest students most. Per-student results and analysis are in Appendix A.5, and other ablations (distilled datasets, hyperparameters) are in Appendix A.

6 Conclusion and Limitations

CCJ shows that a coreset need not consist only of images. By adding just a few scalars (equivalent to an IPC of roughly 10.34 vs 10 on CIFAR-100 and 10.01 vs 10 on ImageNet-1k), we can improve every selector at IPC-10. We also consider the analysis of models' learning behaviour with coresets an equally valuable contribution. The major limitation of our method is that the benefit is primarily in low IPC settings. The stored statistics also draw on all examples, which some privacy settings may not permit. Our proof of Claim 2 holds under squared loss; cross-entropy was confirmed empirically, but a formal proof remains an open question in NTK analysis.

AI USE STATEMENT

In this work, generative AI tools were used to aid in implementing the codebase, editing the writing and writing the proofs. We reviewed all AI-assisted work through PRs and code reviews. We take responsibility for the final content of this work, including text, claims and artefacts produced with the support of generative AI.

ETHICS STATEMENT

This work uses only standard public image-classification benchmarks (CIFAR-10, CIFAR-100, ImageNet-1K) and involves no human subjects or personally identifiable data.

REPRODUCIBILITY STATEMENT

All numbers in this paper are produced by the code which will be released upon paper acceptance. We self-run every numerically compared baseline under the same protocol, and mark cited numbers as such throughout. We tuned hyperparameters using validation splits only.

REFERENCES

Nasir Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, 1974. doi: 10.1109/T-C.1974.223784.

Christopher M. Bishop. Training with noise is equivalent to Tikhonov regularization. *Neural Computation*, 7(1):108–116, 1995.

Blake Bordelon, Abdulkadir Canatar, and Cengiz Pehlevan. Spectrum dependent learning curves in kernel regression and wide neural networks. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1024–1034, 2020. URL https://proceedings.mlr.press/v119/bordelon20a.html.

Abdulkadir Canatar, Blake Bordelon, and Cengiz Pehlevan. Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks. *Nature Communications*, 12:2914, 2021. doi: 10.1038/s41467-021-23103-1.

George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *CVPR*, 2022.

Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir Vapnik. Vicinal risk minimization. In *NeurIPS*, 2001.

Yeseul Cho, Baekrok Shin, Changmin Kang, and Chulhee Yun. Lightweight dataset pruning without full training via example difficulty and prediction uncertainty. In *ICML*, 2025.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. In *ICLR*, 2020.

Ekin D. Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. AutoAugment: Learning augmentation strategies from data. In *CVPR*, 2019.

Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. RandAugment: Practical automated data augmentation with a reduced search space. In *NeurIPS*, 2020.

Justin Cui, Ruochen Wang, Si Si, and Cho-Jui Hsieh. DC-BENCH: Dataset condensation benchmark. In *NeurIPS Datasets and Benchmarks*, 2022.

Terrance DeVries and Graham W. Taylor. Improved regularization of convolutional neural networks with Cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Melanie Ducoffe and Frédéric Precioso. Adversarial active learning for deep networks: A margin based approach. *arXiv preprint arXiv:1802.09841*, 2018.

Chengcheng Guo, Bo Zhao, and Yanbing Bai. DeepCore: A comprehensive library for coreset selection in deep learning. In *International Conference on Database and Expert Systems Applications (DEXA)*, 2022.

Muyang He, Shuo Yang, Tiejun Huang, and Bo Zhao. Large-scale dataset pruning with dynamic uncertainty. In *CVPR Workshops*, 2024.

Katherine L. Hermann, Hossein Mobahi, Thomas Fel, and Michael C. Mozer. On the foundations of shortcut learning. In *ICLR*, 2024.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/5a4be1fa34e62bb8a6ec6b91d2462f5a-Abstract.html.

Arthur Jacot, Berfin Şimşek, Francesco Spadaro, Clément Hongler, and Franck Gabriel. Kernel alignment risk estimator: Risk prediction from training data. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/b367e525a7e574817c19ad24b7b35607-Abstract.html.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://arxiv.org/abs/1902.06720.

Hetian Liu, Jin Cui, Mengcheng Shi, Yanbin Hu, Xinyue Long, Boran Zhao, and Pengju Pen. GLOBE: Trajectory-aligned gradient matching with structured sparse optimization for coreset selection. *arXiv preprint arXiv:2608.02690*, 2026.

Adyasha Maharana, Prateek Yadav, and Mohit Bansal. D^2 pruning: Message passing for balancing diversity and difficulty in data pruning. In *International Conference on Learning Representations (ICLR)*, 2024.

Katerina Margatina, Giorgos Vernikos, Loïc Barrault, and Nikolaos Aletras. Active learning by acquiring contrastive examples. In *EMNLP*, 2021.

Samuel G. Müller and Frank Hutter. TrivialAugment: Tuning-free yet state-of-the-art data augmentation. In *ICCV*, 2021.

Yu-Ichi Ohta, Takeo Kanade, and Toshiyuki Sakai. Color information for region segmentation. *Computer Graphics and Image Processing*, 13(3):222–241, 1980.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Geoff Pleiss, Tianyi Zhang, Ethan R. Elenberg, and Kilian Q. Weinberger. Identifying mislabeled data using the area under the margin ranking. In *NeurIPS*, 2020.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5301–5310, 2019.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *ICLR*, 2018.

Burr Settles. Active learning literature survey. Technical Report 1648, University of Wisconsin–Madison, 2009.

Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://papers.neurips.cc/paper_files/paper/2020/hash/6cfe0e6127fa25df2a0ef2ae1067d915-Abstract.html.

Patrice Simard, Bernard Victorri, Yann LeCun, and John Denker. Tangent prop: A formalism for specifying selected invariances in an adaptive network. In *NeurIPS*, 1992.

James B. Simon, Madeline Dickens, Dhruva Karkada, and Michael R. DeWeese. The eigenlearning framework: A conservation law perspective on kernel regression and wide neural networks. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=FDbQGCAViI.

Charles M. Stein. Estimation of the mean of a multivariate normal distribution. *The Annals of Statistics*, 9(6), November 1981. ISSN 0090-5364. doi: 10.1214/aos/1176345632. URL http://dx.doi.org/10.1214/aos/1176345632.

Haoru Tan, Sitong Wu, Wei Huang, Shizhen Zhao, and Xiaojuan Qi. Data pruning by information maximization. In *International Conference on Learning Representations (ICLR)*, 2025.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. In *ICLR*, 2019.

TorchVision maintainers and contributors. TorchVision: PyTorch's computer vision library. https://github.com/pytorch/vision, 2016.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. In *arXiv preprint arXiv:1811.10959*, 2018.

Yulin Wang, Xuran Pan, Shiji Song, Hong Zhang, Gao Huang, and Cheng Wu. Implicit semantic data augmentation for deep networks. In *NeurIPS*, 2019.

Max Welling. Herding dynamical weights to learn. In *ICML*, 2009.

Shuo Yang, Zeke Xie, Hanyu Peng, Min Xu, Mingming Sun, and Ping Li. Dataset pruning: Reducing training data by examining generalization influence. In *International Conference on Learning Representations (ICLR)*, 2023.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. CutMix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

Xin Zhang, Jiawei Du, Yunsong Li, Weiying Xie, and Joey Tianyi Zhou. Spanning training progress: Temporal dual-depth scoring (TDDS) for enhanced dataset pruning. In *CVPR*, 2024.

Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *ICML*, 2021.

Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. In *WACV*, 2023.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *ICLR*, 2021.

Haizhong Zheng, Rui Liu, Fan Lai, and Atul Prakash. Coverage-centric coreset selection for high pruning rates. In *ICLR*, 2023.

Haizhong Zheng, Elisa Tsai, Yifu Lu, Jiachen Sun, Brian R. Bartoldson, Bhavya Kailkhura, and Atul Prakash. ELFS: Label-free coreset selection with proxy training dynamics. In *ICLR*, 2025.

APPENDIX CONTENTS

A Additional Ablations and Analysis **16**

A.1 Analysis of Gains provided by CCJ and QDA for different coresets 16

A.2 CCJ across Coreset Selection Methods on CIFAR-10 18

A.3 Global Top-k Selection on ImageNet-1K 18

A.4 CCJ in Unlabelled Setting 18

A.5 CCJ across different Student Architectures 20

A.5.1 Analysis behind the difference in the gains across different architectures . . 20

A.6 Does CCJ compose with strong augmentations? 22

A.6.1 Evaluation of Methods with Strong Augmentation and CCJ in CIFAR-100 . 22

A.7 Other Ablations and Baselines 24

A.7.1 Comparison with Fancy PCA and ISDA 24

A.7.2 Different strengths and $K_\star$ in CCJ 24

A.7.3 Different forms of CCJ 25

A.7.4 Comparison with input whitening 25

A.8 How does CCJ work with Dataset Distillation? 26

B Validity of the Assumptions **27**

C Empirical Measurement under Cross Entropy **28**

C.1 In the case of a Linearised ConvNet 28

C.2 In the case of a Nonlinear ConvNet 28

C.3 Other Low-Variance Subspaces 29

D Additional Experiments in the Planted Setting **29**

D.1 CCJ with Linearised ConvNet 29

D.2 CCJ with ConvNet 30

D.3 Preference of the QDA in the Planted Setting 31

E Proofs of Propositions in the Theory **32**

E.1 The linearised model 32

E.2 Proof of Lemma 1 32

E.3 Proof of Lemma 2 33

E.4 Proof of Proposition 3 34

E.5 Proof of Proposition 1 35

E.6 Proof of Proposition 2 37

F Both Claims Under Squared Loss **38**

F.1 Kernel ridge regression on the planted task 38

F.2 The recovered fraction under ridge 39

F.3 Monotonicity lemma 40
F.4 Claim 1 41
F.5 Claim 2 41
F.6 Both claims need a regulariser 42
F.7 What the claims need 43

G Evaluation Protocol **43**

H Compute Cost **44**

A Additional Ablations and Analysis

We measure the preference of a trained network on the held-out conflict examples x_{ab} of Section 3.4. Let p_{hi} and p_{lo} denote the fraction of them predicted as a and b, respectively. The preference of the network is the conflict log-odds $\mathcal{P} = \log \frac{p_{hi}}{p_{lo}}$. Positive $\mathcal{P}$ indicates preference for the high-variance subspace, while negative $\mathcal{P}$ indicates preference for the low-variance subspace, and the measured threshold $t_\star$ is the strength t at which $\mathcal{P} = 0$. Appendix D.3 measures it for the QDA of Section 4.2.

A.1 Analysis of Gains provided by CCJ and QDA for different coresets

In this section, we analyse why different coresets have different levels of improvement when CCJ and QDA is added. The Results are shown in Table 1 and Table 2

At IPC $= 10$, we notice that CCJ improves the most on strong coresets and QDA improves the most on weak ones. At IPC $= 50$, the strong selectors gain very little, while the weak coresets improve more.

The difference in performance improvements due to QDA is straightforward. Students trained on weak coresets have a lot more places where the QDA decision would change the overall decision. The tables for the exact number of decisions which were influenced by QDA is given in Tables 4 and 5.

Table 4: **Decisions changed by QDA on CIFAR-100 at IPC $= 10$.**

selector	changed	wrong→right	right→wrong	wrong→wrong	net
training dynamics					
forgetting	2711	492	70	2149	+422
AUM	3422	658	193	2570	+465
Dyn-Unc	2704	468	74	2161	+394
TDDS	3111	585	125	2401	+460
DUAL	3223	528	174	2521	+354
error and gradient norm					
EL2N	2562	364	49	2149	+315
GraNd	2465	378	51	2036	+327
one-shot uncertainty					
entropy	2368	355	44	1969	+311
least-confidence	2500	415	54	2030	+361
margin	2572	414	58	2100	+356
decision boundary					
CAL	3285	603	170	2512	+433
DeepFool	2754	433	71	2250	+362

To explain the effect of CCJ, we first need to understand the model preference for the different coresets. We propose to test on both unmodified test images and modified test images to understand the extent to which the models prefer the high variance directions of the within-class covariance matrix. We, first describe how we modify the test images. Let μ be the training-set mean and $B \in \mathbb{R}^{K \times D}$ be the top-K eigenvectors of the pooled within-class covariance($K = 20$). A modified test image is $\tilde{x} = \mathrm{clip}\big(\mu + B^\top B(x - \mu)\big)$. It keeps the image's dominant within-class variation (coarse colour, brightness and layout) and discards finer detail. We calculate retention, the fraction of a student's above-chance accuracy that survives on these images: $(a_{\text{modified}} - a_{\text{chance}})/(a_{\text{unmodified}} - a_{\text{chance}})$. We subtract chance to avoid trivial situations. For instance, a useless network is at 1% accuracy in CIFAR-100 on unmodified test images and it also gives 1% accuracy on modified images, we would get 100% retention if we didn't subtract chance.

When we plot the retention for networks trained on AUM, DUAL, EL2N and Forgetting Coresets, the picture is clearly shown.

As shown in Fig 3, CCJ reduces the retention of all the methods. The difference in amount of performance is based on how much the student network can gain from the low-variance subspace.

Table 5: **Decisions changed by QDA on CIFAR-100 at IPC** $= 50$**.**

selector	changed	wrong→right	right→wrong	wrong→wrong	net
training dynamics					
forgetting	2264	570	232	1462	+338
AUM	2401	403	554	1445	−151
Dyn-Unc	2226	567	255	1404	+312
TDDS	2376	501	363	1511	+138
DUAL	2084	291	436	1358	−145
error and gradient norm					
EL2N	2311	552	151	1608	+401
GraNd	2222	527	227	1467	+300
one-shot uncertainty					
entropy	2190	492	201	1497	+291
least-confidence	2012	455	239	1317	+216
margin	2088	451	255	1382	+196
decision boundary					
CAL	2601	471	410	1720	+61
DeepFool	2292	532	279	1480	+253

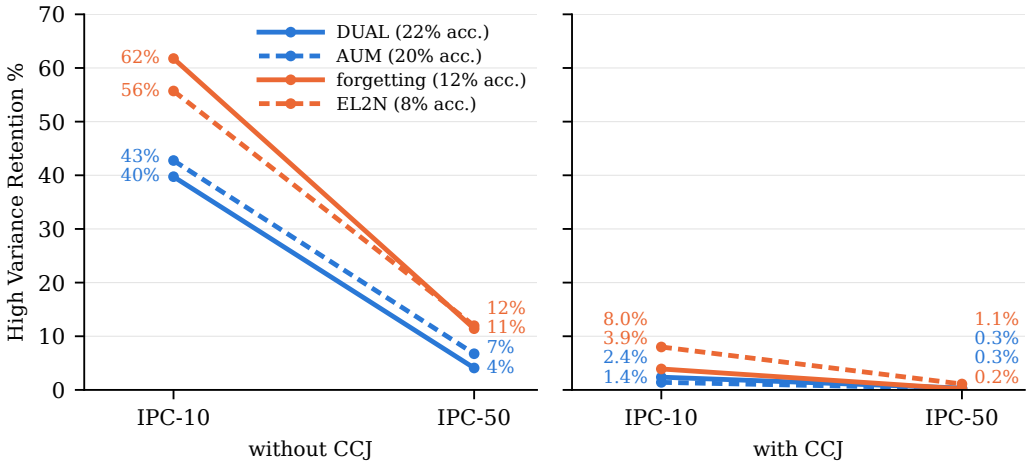


Figure 3: **Retention on modified test images of CIFAR-100 students by coreset.**

To calculate that, we train a network with images whose top-K eigenvectors of the pooled within-class covariance($K = 20$) are replaced by the training-set mean. We make sure to replace the same eigenvectors from the test images when evaluating.

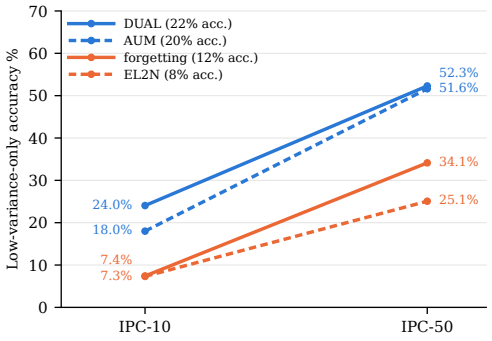


Figure 4: **Accuracy of CIFAR-100 students trained and tested on low-variance-only images.**

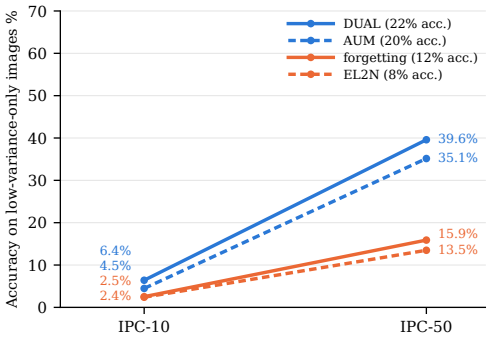


Figure 5: **Accuracy of normally trained CIFAR-100 students on low-variance-only images.**

As shown in Fig 4, there is a lot more useful information that the model can learn from the low-variance subspace for the stronger coresets. At IPC $= 50$, the model already learns the features in the low variance subspace. This can be observed by testing networks trained on standard coresets on test images whose top-K eigenvectors of the pooled within-class covariance($K = 20$) are replaced by the training-set mean.

From Fig 5, we can observe that the slope for the stronger coresets is a lot higher because it learns a lot of the features in the low variance subspace. Thus, the gain we get from CCJ is reduced at IPC-50 for the strong coresets because it already learns the features in the low-variance subspace. We can also see that the students trained using stronger coresets relies less on the high variance subspace at IPC $= 50$ from Fig 3, even in the absence of CCJ, thus they gain less at IPC $= 50$

A.2 CCJ ACROSS CORESET SELECTION METHODS ON CIFAR-10

We repeat Table 1 on CIFAR-10 with the same student, recipe and CCJ settings, over the selectors of Table 2 (Table 6). The IPC-10 columns are trained for 1600 epochs, so that we match the step count of CIFAR-100 at IPC $= 10$. CCJ then improves 8 of the 11 selectors at IPC $= 10$ (mean $+3.20$; AUM is unchanged at $+0.01$, and forgetting and entropy are within one point below `std`), and every selector at IPC $= 50$, by $+1.98$ to $+8.38$ points.

A.3 GLOBAL TOP-k SELECTION ON IMAGENET-1K

Every row of Table 2 is class balanced. In Table 7, each selector uses a global top-k sampler and DUAL uses a β sampler. CCJ was able to improve over DUAL, even when we used β sampling. CCJ improves every selector at both budgets, by $+2.21$ points at IPC $= 10$ and $+1.01$ at IPC $= 50$ on average. The poor performance of DUAL in Table 2 is due to the sampling. Due to poor class coverage, the performance gain of CCJ is lower here than in the class-balanced selection (AUM's IPC-10 coreset covers only 402 of the 1000 classes).

A.4 CCJ IN UNLABELLED SETTING

The training and evaluation protocol behind Table 8 - DINOv2 (Oquab et al., 2024) ViT-S/14 features of the unlabelled training set, spherical k-means into $k = C$ pseudo-classes, ELFS-style selection inside each pseudo-class, and a ConvNet-3 trained for 500 epochs on the selected images under crop, flip and colour jitter. We evaluate across three trainer seeds on one frozen coreset per row

We adapt CCJ such that it doesn't require labels. CCJ's pooled variance is taken over the *pseudo-clusters* instead of the classes.

Table 6: **CCJ across different coreset selection methods on CIFAR-10.** ResNet-18 at 1600 epochs (IPC-10) and 200 epochs (IPC-50).

	IPC-10 (99.8% pruned)				IPC-50 (99% pruned)			
selector	std	+CCJ	std Q	+CCJ Q	std	+CCJ	std Q	+CCJ Q
random	25.60±0.74	32.21±0.94	30.26±0.76	**37.03**±0.80	41.55±0.31	48.55±1.38	44.59±0.15	**52.64**±0.92
training dynamics								
forgetting	22.48±0.28	21.57±0.83	**25.83**±0.18	25.29±0.52	25.74±0.29	27.96±0.98	28.04±0.14	**35.79**±1.50
AUM	22.53±0.37	22.54±0.65	26.90±0.71	**28.51**±0.80	34.50±0.31	36.87±0.56	38.18±0.35	**42.64**±0.70
Dyn-Unc	19.29±0.91	23.10±0.80	24.43±0.47	**28.27**±0.53	26.69±0.41	29.81±0.63	29.82±0.55	**38.14**±0.98
TDDS	30.09±0.34	36.70±1.78	35.66±0.17	**41.85**±1.31	38.03±0.70	45.37±3.01	41.93±0.77	**49.83**±1.60
DUAL	34.39±0.18	42.98±1.04	39.79±0.33	**48.32**±0.98	46.65±0.77	55.03±1.81	49.18±0.49	**57.92**±1.66
error and gradient norm								
EL2N	18.93±0.57	20.15±0.69	22.63±0.52	**24.49**±1.01	24.04±0.81	26.03±0.58	27.50±0.32	**35.18**±3.70
one-shot uncertainty								
entropy	18.25±0.55	18.03±0.57	**20.50**±0.46	20.46±0.59	25.63±0.70	29.14±0.06	28.49±0.58	**36.06**±2.78
least-confidence	16.44±0.49	19.95±0.98	18.78±0.63	**22.76**±1.09	25.71±1.14	30.84±2.15	27.98±1.12	**35.25**±1.47
margin	21.32±1.12	22.94±0.66	24.97±0.85	**27.20**±0.53	32.01±0.87	35.52±2.82	35.59±1.15	**41.97**±1.51
decision boundary								
CAL	30.06±0.57	34.38±0.42	36.01±0.39	**41.18**±0.70	39.45±0.41	46.35±1.50	43.55±0.35	**50.39**±1.40

Table 7: **Global top-k selection in ImageNet-1K.** As Table 2, with each selector's own global top-k in place of a class-balanced coreset; DUAL uses its β sampler.

	IPC-10 (99.2% pruned)				IPC-50 (96.1% pruned)			
selector	std	+CCJ	std Q	+CCJ Q	std	+CCJ	std Q	+CCJ Q
random	10.50±0.20	14.34±0.28	11.83±0.26	**15.74**±0.20	38.32±0.12	**39.72**±0.19	37.98±0.09	39.54±0.27
training dynamics								
forgetting	10.12±0.09	12.46±0.16	11.21±0.08	**13.58**±0.07	37.83±0.21	**38.72**±0.36	37.61±0.28	38.52±0.31
AUM	11.58±0.07	**12.39**±0.05	11.54±0.09	12.35±0.10	29.73±0.09	**29.96**±0.08	28.55±0.12	29.05±0.17
Dyn-Unc	13.26±0.14	15.79±0.15	13.91±0.12	**16.53**±0.10	38.12±0.14	**38.50**±0.07	37.65±0.12	38.18±0.15
TDDS	12.33±0.08	14.84±0.03	13.00±0.06	**15.46**±0.10	35.21±0.07	36.21±0.08	35.01±0.22	**36.28**±0.06
DUAL	19.38±0.15	**20.90**±0.08	18.16±0.19	19.79±0.21	39.45±0.15	**40.03**±0.12	37.17±0.39	38.00±0.29
error and gradient norm								
EL2N	4.05±0.08	5.45±0.04	4.99±0.03	**6.58**±0.04	16.03±0.19	17.55±0.10	17.61±0.16	**19.36**±0.04
one-shot uncertainty								
entropy	4.27±0.07	7.01±0.11	5.27±0.10	**8.24**±0.14	25.25±0.35	27.32±0.13	26.27±0.44	**28.33**±0.13

Table 8: **CCJ with no labels.**

	random		ELFS	
cell	std	+CCJ	std	+CCJ
CIFAR-10 / 1	18.12±0.75	**19.33**±0.20	17.18±0.66	**17.98**±0.43
CIFAR-10 / 10	29.38±0.86	**38.51**±0.75	36.14±0.67	**44.91**±0.36
CIFAR-10 / 50	49.32±0.40	**58.52**±0.45	51.69±0.29	**60.06**±0.63
CIFAR-100 / 1	5.50±0.23	**6.93**±0.09	6.93±0.20	**8.63**±0.05
CIFAR-100 / 10	19.67±0.15	**23.80**±0.27	21.70±0.18	**25.92**±0.26
CIFAR-100 / 50	36.07±0.30	**41.57**±0.24	36.08±0.67	**41.81**±0.11

A.5 CCJ ACROSS DIFFERENT STUDENT ARCHITECTURES

We test whether CCJ improves the performance when used while training different architecture. As seen from Table 3, it does improve the performance of both convolutional and attention-based networks.

Table 9: **CCJ strength per student.** $s^\star$ is the CCJ strength selected per student on a held-out slice of the *training* set.

student	IPC-10	IPC-50
ResNet-18 *(reference)*	1	0.5
ConvNet-3	2	1
ResNet-50	1	1
VGG-11 (BN)	2	0.5
DenseNet-121	1	1
ViT-Tiny/4	1	1

Table 10: **Gain of CCJ alone with s selected for the network.**

	IPC-10		IPC-50	
student	s	gain	s	gain
ResNet-18	1	+7.03	0.5	+0.92
ConvNet-3	1	+4.28	1	+3.86
ResNet-50	1	+3.89	1	+2.82
VGG-11 (BN)	1	+2.30	0.5	+0.71
DenseNet-121	1	+6.55	0.5	+1.55
ViT-Tiny/4	1	+4.35	1	+9.24

A.5.1 ANALYSIS BEHIND THE DIFFERENCE IN THE GAINS ACROSS DIFFERENT ARCHITECTURES

All architectures that we tested prefer the features in the high-variance subspace spanned by top-K eigenvectors of the pooled within-class covariance matrix. It can be shown by measuring the preference values for these models in the planted task (Section 3.1). Figure 6 show the preference at $t = 1.75$. As shown, CCJ reduces the preference in all of the architectures and allows the model to prefer the low-variance feature even at low IPC.

To explain why different architectures show different levels of improvement, we do a similar analysis as Appendix A.1. As shown from Table 11 and Table 12, decisions made by architectures that are not well fitted change a lot with QDA, thus those see a bigger improvement with QDA.

Table 11: **Decisions changed by QDA for each student on CIFAR-100 at IPC $= 10$.**

student	changed	wrong→right	right→wrong	wrong→wrong	net
ResNet-18	3193	522	193	2478	+329
ConvNet-3	3649	626	192	2831	+434
ResNet-50	2943	482	174	2286	+308
VGG-11 (BN)	2895	492	167	2236	+325
DenseNet-121	3265	536	199	2531	+337
ViT-Tiny/4	3669	680	101	2888	+579

Different architectures require different noise scale s. If we choose the noise scale, such that we tune for the best performance when using CCJ and QDA, we could be hurting the network when just using CCJ. This is the case for VGG, setting $s = 2$ makes QDA and CCJ complementary at the expense of reducing the performance when just applying CCJ. The gains for ViT could be because it is undertrained using the standard training recipe. Every student is trained on the same DUAL coreset for 200 epochs at batch size 32 without augmentation, each at its own learning rate tuned on held-out data.

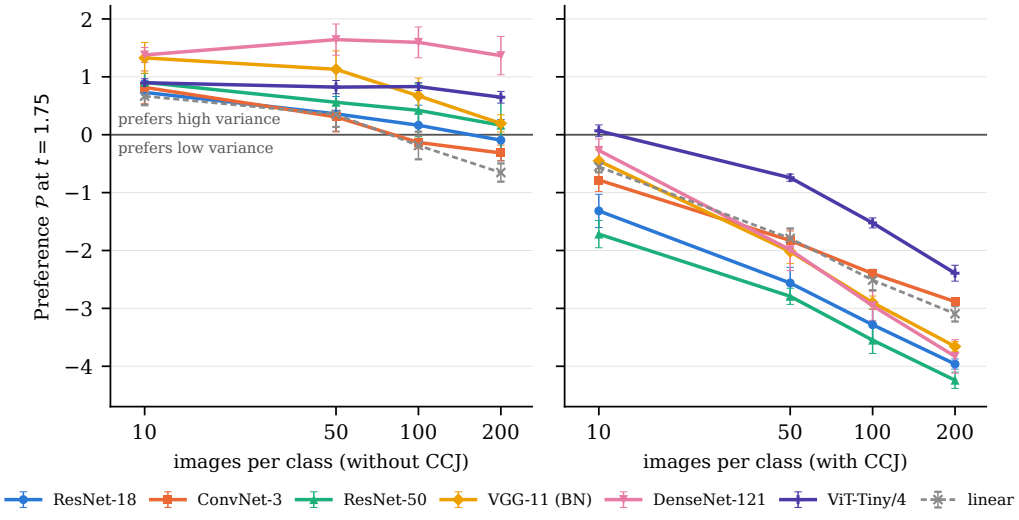


Figure 6: **Preference of each architecture in the planted task at $t = 1.75$.**

Table 12: **Decisions changed by QDA for each student on CIFAR-100 at IPC $= 50$.**

student	changed	wrong→right	right→wrong	wrong→wrong	net
ResNet-18	1946	313	410	1222	−97
ConvNet-3	2521	466	346	1710	+120
ResNet-50	1954	328	330	1296	−2
VGG-11 (BN)	2114	341	373	1399	−32
DenseNet-121	2035	318	438	1279	−120
ViT-Tiny/4	2671	572	193	1906	+379

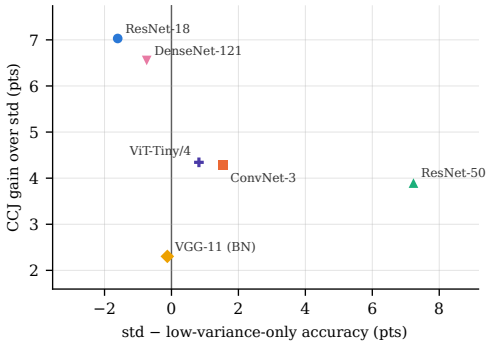


Figure 7: **CCJ gain against std minus low-variance-only accuracy at IPC $= 10$.**

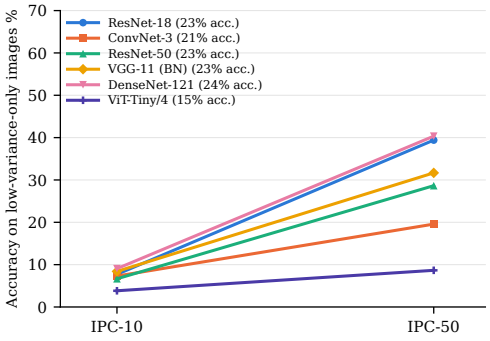


Figure 8: **Accuracy of normally trained CIFAR-100 architectures on low-variance-only images.**

We calculate the difference between the accuracy of a standard trained model on normal inputs and a model trained on low variance images and tested on low variance inputs. This would give us the extent to which a model is able to use the low variance features effectively. From Figure 7, we can see that models whose performance is closer/better to their normally trained counterparts, benefit more with CCJ. We weren't able to explain the behavior VGG. We will explore the interactions of specific components such as residual connections and attention and their connection to feature preference in a later work.

From Figure 8, you can see that based on the slopes of the normally trained networks, that some architectures benefitted greatly by increasing IPC and were able to learn the features in the low variance subspace. The ones whose slope in Figure 8 are the highest, benefits the least from CCJ, as they already have learned the features planted in the low variance subspace. From Figure 9, we can see that at IPC $= 50$ only ViT-Tiny and ConvNet-3 have significant retention on the high variance features, and therefore gains the most with CCJ.

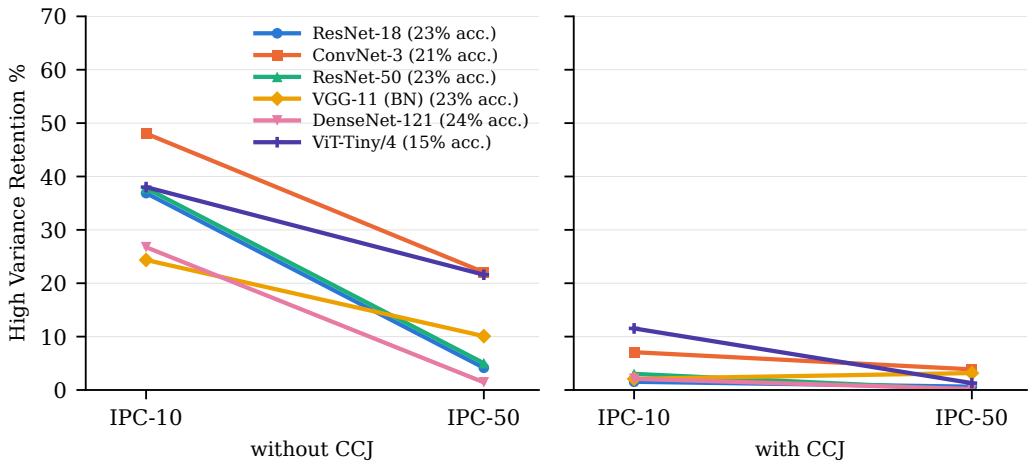


Figure 9: **Retention on modified test images of CIFAR-100 students by architecture.**

A.6 Does CCJ compose with strong augmentations?

Training a coreset under strong augmentation is not standard practice in coreset selection: on CIFAR, selectors are evaluated with random crop and flip only (Cho et al., 2025), which is also our protocol (Table 30). A stronger augmentation could either subsume the jitter or stack with it. Each augmentation is applied at its usual location and configuration. RandAugment (Cubuk et al., 2020), AutoAugment (Cubuk et al., 2019) and TrivialAugment (Müller & Hutter, 2021) are applied on the stored image, Cutout (DeVries & Taylor, 2017) (8×8, zero fill) is applied after normalisation, DSA is applied in pixel space beside the jitter, and Mixup and CutMix on the batch ($\lambda \sim \mathrm{Beta}(1,1)$, CutMix (Yun et al., 2019) at p=0.5). Table 13 adds one augmentation at a time to the DUAL coreset, with the CCJ strength held at s=1 at both IPCs.

A.6.1 Evaluation of Methods with Strong Augmentation and CCJ in CIFAR-100

Table 14 shows the result of every coreset again under RandAugment + Cutout + CutMix, with the CCJ strength set at the value selected on a held-out slice of the training set (s=1 at IPC-10, s=0.5 at IPC-50).

CCJ improves the least where the augmentation already gained the most. Augmentations such as RandAugment, TrivialAugment and AutoAugment already changes most of the samples that CCJ fixes, and that's why the improvement by CCJ on top of them is less. The interesting part is the fact, they act through different mechanisms. In the Planted Setting, (Section 3.4), we can test the trained network on both planted features, and then we evaluate the test performance only when either one is present. In the planted setting, they operate through different mechanisms (Table 15). While

Table 13: **CCJ and QDA under different augmentations.** CIFAR-100, DUAL, ResNet-18 at 200 epochs, CCJ strength s=1 at both budgets, mean±sd over three trainer seeds on one frozen coreset.

	IPC-10 (98% pruned)				IPC-50 (90% pruned)			
augmentation	std	+CCJ	std Q	+CCJ Q	std	+CCJ	std Q	+CCJ Q
crop + flip	22.31±0.16	28.98±0.17	25.83±0.29	**32.39**±0.15	**54.21**±0.33	53.79±0.07	52.89±0.42	54.04±0.11
applied to the stored image, before the jitter								
+ RandAugment	28.98±0.18	31.93±0.54	31.89±0.16	**35.32**±0.44	**55.59**±0.31	54.59±0.42	54.82±0.21	54.97±0.32
+ AutoAugment	27.19±0.54	30.64±0.36	30.86±0.44	**34.45**±0.48	**55.21**±0.38	53.86±0.04	54.80±0.30	54.29±0.12
+ TrivialAugment	27.50±1.02	30.97±0.09	30.91±0.89	**34.81**±0.17	**55.39**±0.37	54.81±0.16	54.63±0.31	54.89±0.12
applied to the batch, after the jitter								
+ Cutout	23.49±0.82	29.20±0.67	26.94±0.57	**32.42**±0.59	**55.41**±0.09	54.24±0.32	54.16±0.05	54.63±0.21
+ Mixup	25.79±0.66	30.40±0.78	29.04±0.41	**32.66**±0.41	52.64±0.26	**53.42**±0.14	49.63±0.34	50.62±0.28
+ CutMix	24.03±0.16	31.40±0.41	27.95±0.20	**33.93**±0.43	54.62±0.32	**55.45**±0.08	52.28±0.45	53.77±0.16
+ DSA	24.83±0.54	30.56±0.53	27.97±0.46	**34.13**±0.38	**54.75**±0.24	54.04±0.46	53.89±0.12	54.45±0.46
+ RandAugment + Cutout + CutMix	30.51±0.89	32.00±0.97	32.57±0.65	**34.57**±0.64	**56.24**±0.33	55.00±0.51	53.83±0.29	53.88±0.47

Table 14: **CCJ and QDA across selectors, under strong augmentation.** CIFAR-100, ResNet-18 at 200 epochs under RandAugment + Cutout + CutMix, mean±sd over three trainer seeds on one frozen coreset per row.

	IPC-10 (98% pruned)				IPC-50 (90% pruned)			
selector	std	+CCJ	std Q	+CCJ Q	std	+CCJ	std Q	+CCJ Q
training dynamics								
forgetting	14.93±0.53	15.09±0.89	20.83±0.58	**22.09**±0.80	43.28±0.08	45.03±0.34	45.81±0.10	**47.76**±0.43
AUM	28.47±0.85	28.24±2.11	32.34±0.67	**32.81**±1.64	57.13±0.19	**57.71**±0.23	54.65±0.51	55.24±0.29
Dyn-Unc	15.32±0.69	16.26±1.49	21.02±0.54	**23.45**±1.28	45.01±0.35	46.20±0.43	47.62±0.08	**49.26**±0.24
TDDS	23.82±0.92	24.53±0.52	29.11±0.70	**30.63**±0.43	51.82±0.21	52.64±0.25	51.86±0.05	**53.06**±0.20
DUAL	30.80±0.08	32.21±0.66	32.93±0.20	**34.56**±0.61	56.51±0.19	**56.77**±0.20	54.08±0.29	54.61±0.20
error and gradient norm								
EL2N	11.21±0.47	13.44±0.84	16.02±0.71	**20.27**±0.36	33.02±0.57	34.90±0.75	37.18±0.29	**38.89**±0.33
GraNd	13.40±0.40	14.59±0.91	17.98±0.34	**20.63**±0.52	44.50±0.51	45.29±0.03	46.61±0.39	**47.76**±0.28
one-shot uncertainty								
entropy	13.57±0.45	13.11±1.65	18.16±0.57	**19.52**±0.83	43.84±0.53	45.28±0.26	46.19±0.46	**47.58**±0.33
least-confidence	14.20±0.13	12.90±0.77	18.84±0.17	**19.87**±0.51	47.50±0.37	47.45±0.22	49.06±0.25	**49.07**±0.07
margin	14.65±0.59	14.91±2.30	19.24±0.69	**21.57**±1.47	49.05±0.30	49.99±0.39	50.05±0.34	**51.23**±0.27
decision boundary								
CAL	24.50±0.65	26.30±1.41	28.73±0.54	**30.69**±1.31	48.68±0.28	**49.61**±0.42	48.26±0.20	49.01±0.27
DeepFool	15.84±0.37	16.08±0.87	20.97±0.45	**22.70**±0.43	48.42±0.34	49.01±0.40	49.50±0.41	**50.26**±0.45
coverage								
CCS	30.03±1.72	31.50±1.04	32.36±1.06	**34.39**±0.83	53.78±0.20	**54.82**±0.17	51.27±0.18	52.59±0.21
D^2	29.58±0.63	31.36±1.30	33.11±0.41	**34.91**±0.95	56.13±0.54	**56.93**±0.08	54.50±0.60	55.09±0.19
Herding	22.56±0.45	20.44±4.42	**27.49**±0.36	27.02±2.96	53.98±0.25	**55.09**±0.36	53.43±0.29	54.77±0.15

CCJ increases the performance when only testing on images with just low-variance only feature, the performance stays the same in the case of RandAugment.

Table 15: **What CCJ and RandAugment each do to the planted features.**

	IPC-10			IPC-200		
training	high-var. only	low-var. only	both	high-var. only	low-var. only	both
no augmentation	67.5±4.9	21.0±1.6	79.1±2.1	95.0±1.3	48.5±0.7	99.6±0.2
+CCJ	63.3±5.8	**37.6**±1.4	86.7±3.4	84.2±3.3	**88.8**±0.7	99.8±0.1
crop + flip	55.1±7.1	17.7±2.0	67.5±4.1	88.7±4.5	24.4±4.1	96.7±0.7
crop + flip + RandAugment	48.1±5.3	17.7±0.9	60.8±2.6	85.1±5.7	25.4±4.3	96.4±0.7

On CIFAR-100, we can see why the CCJ improves little over RandAugment. We apply the same test to the DUAL coreset with the high-variance-only and low-variance-only test images of Appendix A.1 (Table 16). RandAugment also moves the network away from the high-variance subspace. At IPC-10 it lowers retention from 39.7% to 15.0% and raises low-variance-only accuracy from 6.4% to 16.4%. At a similar accuracy (29.6% against 28.3%), CCJ goes much further, to 2.4% retention and 27.0% low-variance-only accuracy. The planted task therefore exaggerates the difference between the two.

Table 16: **What CCJ and RandAugment each do to a CIFAR-100 student.**

	IPC-10			IPC-50		
training	accuracy	retention	low-var. only	accuracy	retention	low-var. only
crop + flip	22.40±0.80	39.74±7.64	6.41±0.39	53.97±0.33	4.09±0.80	39.59±1.11
+CCJ	29.60±1.01	2.37±0.99	26.95±1.29	53.93±0.38	0.35±0.44	52.45±0.41
+ RandAugment	28.26±0.40	15.03±2.25	16.36±0.73	55.85±0.29	4.00±0.19	44.34±0.81
+ RandAugment +CCJ	30.67±0.91	2.13±0.37	28.70±0.65	54.49±0.27	0.36±0.60	52.97±0.37

A.7 Other Ablations and Baselines

A.7.1 Comparison with Fancy PCA and ISDA

We compare CCJ with two augmentations that also use data statistics: the Fancy PCA augmentation of Krizhevsky et al. (2012), which shifts every pixel of an image along the principal components of the RGB covariance, and ISDA (Wang et al., 2019), which augments the deep features along class-conditional covariances estimated online. Both are evaluated using their published settings, σ=0.1 for Fancy PCA and λ_0=0.5 for ISDA, and we compute the RGB covariance on the full training set. CCJ composes with ISDA, and ISDA +CCJ is the best on all six coresets at IPC = 10

Table 17: **Fancy PCA and ISDA against CCJ, IPC-10.**

selector	std	+CCJ	+Fancy PCA	+ISDA	Fancy PCA +CCJ	ISDA +CCJ
DUAL	23.22±0.08	29.27±0.62	23.53±0.65	24.16±0.72	29.10±1.08	**31.54**±0.42
AUM	20.06±0.49	26.26±0.25	21.56±0.62	21.97±0.17	26.37±0.30	**28.36**±0.60
CAL	18.50±0.45	23.44±1.41	19.33±0.26	19.41±0.51	22.46±0.85	**25.77**±0.13
TDDS	15.98±0.44	20.69±0.37	16.44±0.30	16.10±0.38	21.42±0.62	**22.50**±0.65
EL2N	8.27±0.49	10.87±0.40	8.61±0.35	8.22±0.23	11.10±0.16	**11.44**±0.59
entropy	10.68±0.14	12.85±0.84	11.43±0.15	11.26±0.19	12.90±0.24	**13.19**±0.34

A.7.2 Different strengths and $K_\star$ in CCJ

Table 19 varies the knee $K_\star$ of Eq. 11 with the strength held at $s = 1$, for ResNet-18 students trained on DUAL coresets. At IPC = 10 the knee doesn't affect the performance by a lot, and only the two ends lose accuracy. At IPC = 50 accuracy falls steadily as the knee widens. This is due to the fact that the strength of the noise is too high and as you increase the knee, the noise affects more directions.

Table 18: **Fancy PCA and ISDA against CCJ, IPC-50.**

selector	std	+CCJ	+Fancy PCA	+ISDA	Fancy PCA +CCJ	ISDA +CCJ
DUAL	53.85±0.69	53.79±0.15	54.94±0.86	**55.47**±0.31	53.69±0.26	54.33±0.08
AUM	52.41±0.85	53.69±0.27	53.78±0.38	54.36±0.44	53.41±0.39	**54.64**±0.25
CAL	43.39±0.43	46.13±0.47	43.97±0.33	44.91±0.32	46.12±0.40	**47.10**±0.50
TDDS	46.76±1.40	46.91±0.76	46.34±1.28	**48.09**±0.88	46.84±0.51	47.89±0.42
EL2N	24.10±0.44	28.37±0.25	25.34±0.83	25.34±0.23	27.81±0.49	**28.83**±0.35
entropy	35.51±1.67	40.05±0.55	37.76±1.14	36.04±1.02	40.44±0.16	**41.59**±0.72

Table 19: **Knee $K_\star$ of CCJ at $s = 1$ on CIFAR-100.**

$K_\star$	IPC = 10	IPC = 50
no jitter	22.35	54.44
10	28.16±0.29	**54.63**±0.19
20	29.45±0.35	54.47±0.22
40	**29.53**±0.57	54.33±0.14
80	29.10±0.32	53.78±0.14
160	29.12±0.40	52.89±0.22
320	27.78±0.87	51.29±0.29
640	26.77±0.69	48.77±0.37

Table 20: **Strength s of CCJ at $K_\star = 80$ on CIFAR-100.**

s	IPC = 10	IPC = 50
0 (no jitter)	22.77±0.75	54.26±0.10
0.25	24.51±0.63	55.06±0.37
0.5	27.37±0.57	**55.17**±0.30
1	**29.32**±0.53	53.95±0.14
1.5	29.05±0.59	52.28±0.25
2	27.05±0.94	50.56±0.40
3	19.79±1.81	45.97±0.35

Table 20 varies s at $K_\star = 80$ under the protocol of Table 19. We can see that the best strength s for IPC $= 10$ is $s = 1$ and at IPC $= 50$, the best is $s = 0.5$. Table 21 shows the share of variance in the top-$K_\star$ DCT atoms in CIFAR-100.

A.7.3 Different forms of CCJ

Table 22 compares CCJ with three other forms of the jitter on DUAL coresets, under the same setting of Table 1. The first is isotropic Gaussian noise, $x \leftarrow x + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ in pixel space, clipped to the image range as CCJ is. We set its total variance $\mathbb{E}\|\varepsilon\|^2$ to $1/16$, $1/4$ and 1 times that of CCJ at $s = 1$ and choose among them on a held-out slice of the training set at IPC $= 10$. The chosen level, $1/4$, is used at both IPCs. The second is the hard knee, the $\beta \to \infty$ limit of Eq. 11, which jitters the top $K_\star = 80$ atoms at their full standard deviation and leaves the rest untouched. At the same s the jitter of the hard knee has a larger total variance than that of the soft one, so we also run it with s scaled down to match the total variance of the soft knee. The third keeps the top 80 atoms but gives each of them the same standard deviation, the root mean square of theirs, so the noise is isotropic inside those atoms and has the total variance of the hard knee at the same s.

Isotropic noise does not improve on `std`. The hard knee is close to the soft knee. The scale of each atom does matter. Isotropic noise in the top 80 atoms performs worse than the hard knee and the soft knee at the same total variance. In terms of storage, the hard knee stores less than the soft knee.

A.7.4 Comparison with input whitening

By Proposition 3, the availability of eigendirection j is $\lambda_j g_j$, and whitening the input sets the data term λ_j to one. We test whether a fixed whitening can replace CCJ, on DUAL coresets at IPC $= 10$ under the setting of Table 22. Whitening changes what the network sees, so we apply it to the test images as well as the training images. The first form uses the DCT basis of Section 4.1 and scales atom a by $(\sigma_a^2 + \epsilon)^{-\alpha/2}$, with σ_a^2 as in Eq. 11 and ϵ equal to 0.1 times the mean of the σ_a^2. Setting $\alpha = 1$ whitens the atoms and $\alpha = 1/2$ half-whitens them. The second form is the ZCA whitening

Table 21: **Share of within-class variance in the top-$K_\star$ DCT atoms, CIFAR-100.**

$K_\star$	10	20	40	80	160	320	640
share	0.633	0.725	0.805	0.870	0.923	0.962	0.986

Table 22: **Different forms of CCJ on DUAL coresets, CIFAR-100.**

form	IPC = 10	IPC = 50
no jitter (`std`)	22.53±0.53	54.06±0.39
isotropic Gaussian	21.33±0.53	44.62±0.44
soft knee, $s = 1$ (Table 1)	29.46±0.16	54.02±0.08
hard knee, variance of soft $s = 1$	29.24±0.47	53.99±0.06
isotropic in top-80, variance of soft $s = 1$	25.94±1.36	51.53±0.18
hard knee, $s = 1$	**30.05**±0.19	53.62±0.35
soft knee, $s = 0.5$	27.34±0.29	55.20±0.47
hard knee, variance of soft $s = 0.5$	28.02±0.49	**55.32**±0.22
isotropic in top-80, variance of soft $s = 0.5$	26.14±0.09	54.15±0.25

with $\epsilon = 0.1$ that trajectory matching applies to CIFAR (Cazenavette et al., 2022), computed from the full covariance of the training set.

Whitening with $\alpha = 1$ reaches 27.68, 5.15 points above `std` and 1.78 below CCJ, (Table 23). It recovers 74% of the gain of CCJ, against 65% for ZCA and 52% for half-whitening.

Table 23: **Input whitening against CCJ on DUAL coresets, CIFAR-100, IPC = 10.**

input	IPC = 10
no jitter (`std`)	22.53±0.53
DCT whitening, $\alpha = 1$	27.68±0.49
DCT whitening, $\alpha = 1/2$	26.11±0.50
ZCA (Cazenavette et al., 2022)	27.04±0.82
CCJ, soft knee, $s = 1$	**29.46**±0.16

A.8 How does CCJ work with Dataset Distillation?

Table 24: **CCJ and QDA on DC-BENCH distilled sets.**

set	`std`	+CCJ	`std` Q	+CCJ Q
IPC-10				
random *(real)*	18.69±0.26	19.61±0.04	23.74±0.08	**24.59**±0.05
DC (Zhao et al., 2021)	25.19±0.29	26.11±0.16	28.16±0.27	**28.96**±0.19
DSA (Zhao & Bilen, 2021)	32.24±0.07	33.07±0.12	33.92±0.22	**34.66**±0.07
DM (Zhao & Bilen, 2023)	29.83±0.11	30.91±0.22	33.04±0.08	**33.94**±0.42
TM (Cazenavette et al., 2022)	38.05±0.39	39.00±0.23	39.43±0.10	**40.09**±0.13
IPC-50				
random *(real)*	34.68±0.33	36.44±0.09	37.07±0.34	**38.78**±0.16
DC	30.05±0.21	30.53±0.17	32.38±0.22	**32.78**±0.22
DSA	43.14±0.43	43.96±0.29	43.82±0.25	**44.59**±0.17
DM	43.19±0.12	43.44±0.46	44.21±0.12	**44.63**±0.20
TM	46.34±0.13	46.56±0.11	46.38±0.08	**46.89**±0.14

We test whether our method improves the datasets generated through Dataset Distillation methods (Wang et al., 2018). We evaluate on the four published CIFAR-100 sets of DC-BENCH (Cui et al., 2022) and its own real-image `random` baseline, at IPC = 10 and IPC = 50. For each method, the corresponding student model follows the original training recipe, suggested by the respective authors. Because each set keeps its own recipe, these accuracies are not comparable with the main tables: the same real `random` coreset reads 18.69 here at IPC-10 and 15.01 in Table 1.

Table 25 takes the high-variance subspace as the top 20 eigenvectors of the training covariance and the low-variance subspace as its complement, and trains all five IPC-10 sets with one recipe (a ConvNet for 500 epochs with standard augmentation, three seeds), so its accuracies are not comparable with Table 24. *High-variance removed* is a network trained on full images and tested with the

high-variance subspace projected out, *low-variance only* is a network trained and tested on the low-variance projection, *surplus* is low-variance only minus high-variance removed, and *high-variance contribution* is full minus low-variance only.

Table 25: **Where each set keeps its class signal.**

set	full	high-variance removed	low-variance only	surplus	high-variance contribution
random *(real)*	19.46	5.26	16.83	**+11.57**	**2.63**
DC (Zhao et al., 2021)	27.71	12.64	15.21	+2.57	12.50
DSA (Zhao & Bilen, 2021)	30.99	13.91	19.18	+5.27	11.81
DM (Zhao & Bilen, 2023)	27.97	14.67	20.20	+5.53	7.77
TM (Cazenavette et al., 2022)	35.37	19.66	26.58	+6.92	8.79

B VALIDITY OF THE ASSUMPTIONS

The results of Section 3 are derived under Assumptions 1 and 2. We discuss each in turn: what it is used for, when it holds, and how far the experiments of Section 3.4 satisfy it. Those experiments train a nonlinear ConvNet on real CIFAR-10 backgrounds, so they test the conclusions without imposing any of these assumptions.

Assumption 1(a) assumes Gaussian backgrounds but natural images are not Gaussian. It is needed in two places. One, for ρ to be exactly the weight of the Bayes-optimal classifier (Lemma 1), and for Stein's identity, which the proofs of Propositions 1 and 3 use to turn expectations over the backgrounds into derivatives of the kernel. For non-Gaussian backgrounds these two steps become approximations.

Assumption 1(b) assumes that the kernel is class diagonal. For each pair of images the tangent kernel is a $C \times C$ matrix whose entry (c, c') is the inner product of the parameter gradients of logits c and c'. When each class has its own output weights, the off-diagonal entries average to zero over the random initialisation and the diagonal entries coincide, and both become exact as the width grows (Jacot et al., 2018). At finite width they hold only approximately. The assumption lets every result be written with the single scalar kernel Θ.

Assumption 1(c) assumes that the kernel is twice differentiable. It is needed for the second-order expansion of the kernel in the planted shifts and for Stein's identity. It holds for networks with smooth activations.

Assumption 1(d) assumes a uniform and centred logits at initialisation. They give a conflict margin of zero before training. This holds in expectation over a random initialisation with zero-mean output weights, and approximately for a single wide network whose initial logits are small.

Assumption 1(e) assumes the strength of the planted features are small. Keeping only the lowest-order term in their scale l is what yields the decomposition of the margin in Eq. 34, and makes the threshold $t_\star$ independent of l. The approximation is accurate when l is small compared with one standard deviation of the backgrounds along the planted directions, since l is the whitened length of the high-variance feature. The experiments do not satisfy this condition as the planted ConvNet runs use $l = 3$, three standard deviations. They therefore test whether the conclusions survive outside the regime in which they are derived. The $\approx$ in Propositions 1 and 2 marks this truncation: their proofs drop remainders of relative order l (Eqs. 34 and 40), bounded uniformly over any finite training time.

Assumption 2 states that the whitened coordinates are eigenfunctions of the kernel, that is, that the kernel is diagonal in the eigenbasis of Σ. Propositions 1 and 2 use it. It does not hold exactly for a real network.

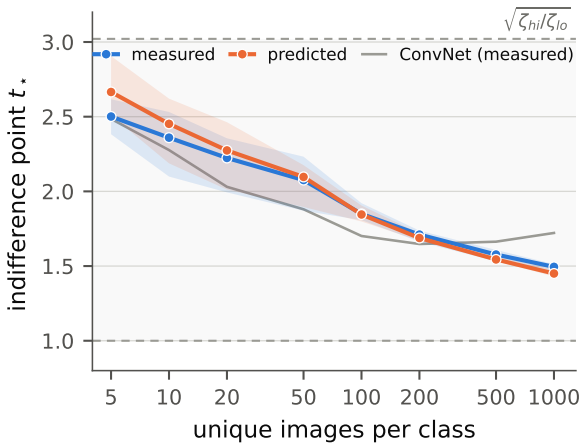


Figure 10: **The threshold $t_\star$ against IPC in the linearised ConvNet, measured and predicted.**

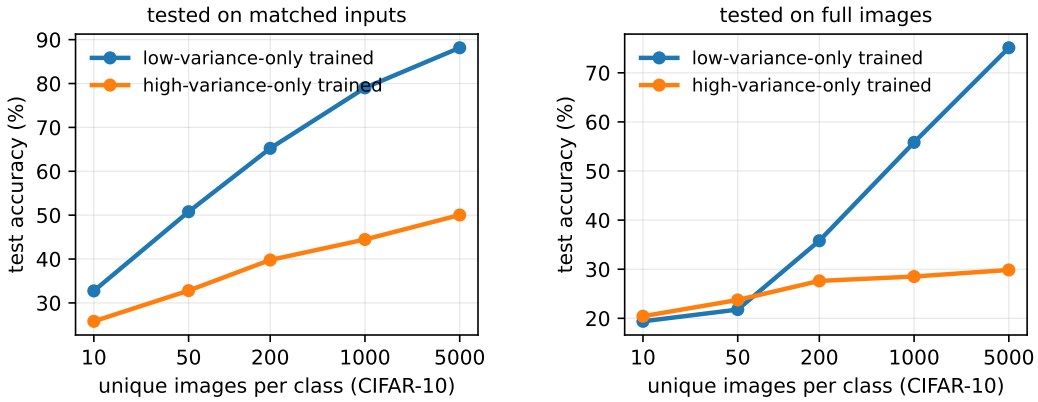


Figure 11: **High-variance-only and low-variance-only trained ConvNets on CIFAR-10.**

C Empirical Measurement under Cross Entropy

C.1 In the case of a Linearised ConvNet

We perform the analysis of Section 3.4 on the linearised ConvNet of Appendix E.1, trained with the same planted data, optimiser and schedule as the nonlinear ConvNet.

The availability Figure 1 would be the same for both the linearised and nonlinear model as it is calculated at initialisation. At initialisation $u = 0$, so the linearised model has the same outputs and the same tangent kernel Θ_0 as the ConvNet it is built from, and ζ_j and g_j are functions of that kernel alone. Figure 1 is therefore also the availability of the linearised ConvNet.

Figure 10 shows the measured and predicted thresholds. The measured $t_\star$ falls with IPC over the whole range, unlike in the case of a nonlinear ConvNet. The nonlinear ConvNet threshold $t_\star$ reduces up to about 200 images per class and then rises.

C.2 In the case of a Nonlinear ConvNet

Our analysis in Section 3.4, is under the premise that in natural images the low variance features are more predictive. We test it by training a ConvNet only on the low-variance subspace and another only on the high-variance subspace. Figure 11 shows that the low-variance one is more predictive.

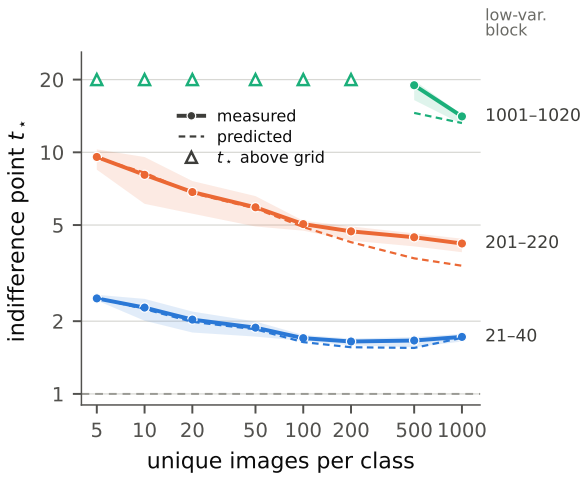


Figure 12: **The threshold $t_\star$ against IPC for three low-variance subspaces, measured and predicted.**

Table 26: **Recovered fractions and the threshold in the linearised ConvNet, with and without CCJ.**

	no CCJ					CCJ				
IPC	L_{hi}	L_{lo}	L_{hi}/L_{lo}	$t_\star$ pred.	$t_\star$ meas.	L_{hi}	L_{lo}	L_{hi}/L_{lo}	$t_\star$ pred.	$t_\star$ meas.
5	0.429	0.063	6.77	2.665	2.501	0.371	0.128	2.89	1.861	1.821
10	0.507	0.094	5.41	2.452	2.360	0.423	0.190	2.22	1.617	1.620
20	0.545	0.127	4.28	2.274	2.224	0.450	0.230	1.95	1.498	1.530
50	0.633	0.182	3.48	2.097	2.076	0.498	0.324	1.54	1.317	1.343
100	0.684	0.242	2.82	1.845	1.849	0.511	0.408	1.25	1.155	1.182
200	0.704	0.295	2.39	1.688	1.711	0.513	0.473	1.08	1.053	1.104
500	0.698	0.341	2.05	1.544	1.577	0.496	0.517	0.96	0.973	1.037
1000	0.675	0.370	1.83	1.450	1.495	0.480	0.533	0.90	0.932	1.012

C.3 Other Low-Variance Subspaces

In the default planted setting, we consider eigendirections 21–40 as the low-variance subspace. Figure 12 repeats the threshold measurement of Section 3.4 with the low-variance subspace moved deeper, to eigendirections 201–220 and 1001–1020. As the difference in variance grows, the threshold rises at every IPC. We weren't able to measure the indifference point, if low variance feature is planted in 1001–1020. At both depths a low-variance feature must be far more predictive than its high-variance competitor before the model follows it, and the factor it needs falls as IPC grows, as Claim 2 states.

D Additional Experiments in the Planted Setting

D.1 CCJ with Linearised ConvNet

In this section, we explore the effects of applying CCJ to the linearised ConvNet of Appendix C.1 The jitter follows Eq. 11 with $K_\star = 20$, $\beta = 1$ and $s = 1$. In the planted task the frame is the eigenbasis of the backgrounds rather than the DCT basis. The jitter is applied to the training images only.

Figure 13 compares the thresholds with and without CCJ, and Table 26 lists the values. CCJ lowers the measured $t_\star$ at every IPC. The threshold predicted from these fractions by Eq. 7 is within 0.08 of the measured one at every IPC.

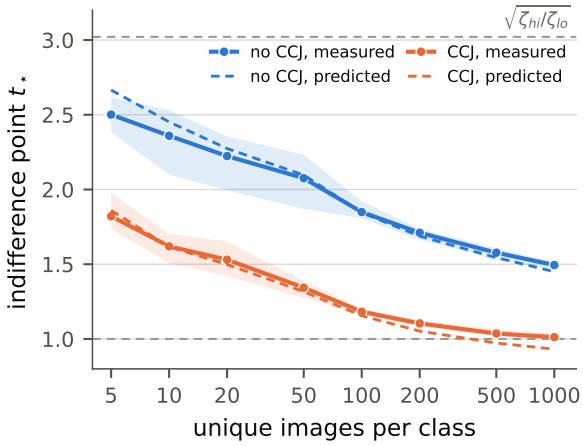


Figure 13: **The threshold $t_\star$ against IPC in the linearised ConvNet, with and without CCJ.**

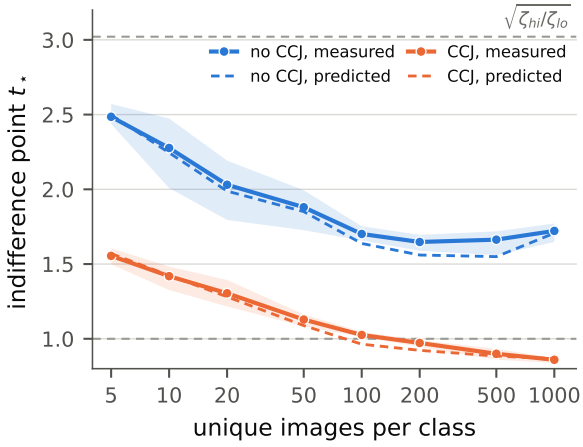


Figure 14: **The threshold $t_\star$ against IPC in the ConvNet, with and without CCJ.**

D.2 CCJ WITH CONVNET

We perform the same analysis of Appendix D.1 with the nonlinear ConvNet of Section 3.4. CCJ weakens the preference for the high-variance feature at every IPC. At equal predictivity ($t = 1$) it lowers the preference $\mathcal{P}$ of Appendix A from $+2.10$ to $+1.07$ at ten images per class, and from $+3.15$ to -0.66 at a thousand. From 200 images per class upwards $\mathcal{P}$ is negative, so the network trained with CCJ follows the low-variance feature even when the two features are equally predictive on clean images.

Figure 14 compares the thresholds, and Table 27 lists the values. CCJ lowers the measured $t_\star$ at every IPC, and unlike in the linearised network it takes the threshold below one.

The fall in predictivity alone is too small to explain the shift in $t_\star$. Averaged over each subspace, the jitter reduces the whitened strength of the high-variance feature to 0.82 of its value and that of the low-variance feature to 0.94. If CCJ acted only through predictivity, it would divide $t_\star$ by the ratio of the two, 1.145, giving 1.99, 1.49 and 1.50 at ten, a hundred and a thousand images per class, against the measured 1.42, 1.03 and 0.86. It would also lower L_{lo}, whereas CCJ raises L_{lo} by a factor of 1.8 to 2.9 (Table 27).

The reduction in $t_\star$ is made during training. To measure the availability at the end of training, we retrain the networks at $t = 1$ with ten and a thousand images per class and replace Θ in Eq. 3 by the tangent kernel of the trained network, summed over the logits after subtracting their mean, since the cross-entropy does not depend on the mean logit. We evaluate it on clean backgrounds the network

Table 27: **Recovered fractions and the threshold in the ConvNet, with and without CCJ.**

	no CCJ					CCJ				
IPC	L_{hi}	L_{lo}	L_{hi}/L_{lo}	$t_\star$ pred.	$t_\star$ meas.	L_{hi}	L_{lo}	L_{hi}/L_{lo}	$t_\star$ pred.	$t_\star$ meas.
5	0.396	0.069	5.73	2.498	2.485	0.304	0.130	2.33	1.570	1.555
10	0.505	0.106	4.78	2.245	2.277	0.368	0.195	1.89	1.424	1.419
20	0.538	0.144	3.74	1.988	2.031	0.393	0.259	1.52	1.281	1.305
50	0.600	0.187	3.20	1.851	1.880	0.399	0.347	1.15	1.089	1.129
100	0.548	0.212	2.58	1.638	1.701	0.373	0.392	0.95	0.964	1.026
200	0.471	0.201	2.34	1.560	1.648	0.343	0.387	0.89	0.923	0.972
500	0.360	0.151	2.38	1.550	1.663	0.267	0.321	0.83	0.884	0.900
1000	0.334	0.111	3.00	1.706	1.722	0.264	0.320	0.82	0.866	0.859

Table 28: **Availability in the ConvNet with and without CCJ.**

	CCJ / no CCJ		ζ_{hi}/ζ_{lo}	
	ζ_{hi}	ζ_{lo}	no CCJ	CCJ
initialisation	$1.19_{\pm 0.01}$	$0.89_{\pm 0.01}$	$9.15_{\pm 0.53}$	$12.18_{\pm 0.85}$
trained, IPC 10	$1.72_{\pm 0.28}$	$2.02_{\pm 0.25}$	$10.24_{\pm 0.82}$	$8.73_{\pm 0.96}$
trained, IPC 1000	$0.063_{\pm 0.016}$	$0.22_{\pm 0.06}$	$15.91_{\pm 2.16}$	$4.43_{\pm 0.13}$

was not trained on. Without CCJ, training makes the high-variance subspace more available still: ζ_{hi}/ζ_{lo} rises from about 9.1 at initialisation to 10.24 at ten images per class and 15.91 at a thousand. With CCJ it ends below its initial value, at 8.73 and 4.43. At a thousand images per class the high-variance availability of the network trained with CCJ is 6% of that of the network trained without it. Much of this is a smaller kernel overall (its mean diagonal is 12% of the other's), but the high-variance availability falls about twice as far. At ten images per class, CCJ lowers the availability of the high-variance subspace relatively more compared to the other directions.

Most of the rise in L_{lo} comes from the residual that CCJ leaves on the training images. Under Assumption 2, each step of gradient descent changes L_q in proportion to the correlation between the residual $e_y - \mathrm{softmax}(f)$ on the training images and their whitened coordinate along ν_q^c, weighted by the availability of each direction, as in the argument of Lemma 2 applied at every step. We record this residual along training, on jittered images for the network trained with CCJ. CCJ slows the fit to the coreset: summed over training, the residual is 2.0, 2.2 and 3.7 times larger with CCJ at ten, a hundred and a thousand images per class, although both networks fit the coreset exactly by the end. L_{lo} with CCJ overtakes L_{lo} without it within the first 11 steps, as the network without CCJ begins to fit its coreset, and stays ahead until the end.

The same accounting shows how the lower predictivity acts on the high-variance feature. The jitter along ν_{hi}^c is anticorrelated with the residual: an image jittered towards its own class has a smaller residual, and one jittered away from it a larger one. This makes the contribution of the jitter and background to the high-variance sum 3.8, 3.1 and 6.0 times more negative. The accounting starts from the kernel at initialisation, and it does not predict the size of the fall in L_{hi}.

D.3 Preference of the QDA in the Planted Setting

We measure the preference of the quadratic discriminant of Section 4.2, in the setting of Section 3.4. Each QDA works in a rank-20 frame $B \in \mathbb{R}^{D\times 20}$, and its class means and covariances are fitted on all $50{,}000$ planted CIFAR-10 training images, carrying both features. We then score $5{,}000$ conflict examples x_{ab} built on CIFAR-10 test images and report the preference $\mathcal{P}$ of Appendix A. QDA strongly prefers the high-variance feature, with $\mathcal{P} = +3.28$.

Table 29 compares different QDA subspaces.First, the 20 DCT atoms of highest pooled within-class variance, the one used in the paper. Second, the top 20 eigenvectors of the background covariance and a few more. We see that our method has the most overlap with the high variance subspace, after that of using the exact top 20 eigenvectors. The order in which the atoms are chosen matters. Ranking them by their Fisher ratio, which measures predictivity rather than variance, admits atoms

Table 29: **Behaviour of QDA in the planted setting.**

frame	$\mathcal{P}$	conflict predicted as (%)		overlap with		clean acc. (%)
		a (high)	b (low)	P_{hi}	P_{lo}	
DCT, variance order (ours)	$+3.28_{\pm 0.33}$	81.0	3.1	0.845	0.125	87.1
top-20 eigenvectors	$+4.18_{\pm 0.24}$	88.2	1.4	1.000	0.000	88.2
DCT, Fisher order	$+0.78_{\pm 0.28}$	49.7	22.9	0.487	0.337	90.6
DCT, lowest frequencies	$+2.36_{\pm 0.28}$	62.1	5.9	0.521	0.068	72.2
random subspace	$+1.61_{\pm 0.12}$	52.5	10.5	0.006	0.006	73.0

that carry the low-variance feature (overlap 0.337) and lowers the preference to $+0.78$, below the random subspace, even though that frame is the most accurate classifier on clean inputs. We use the variance order so that the QDA reads the high-variance subspace that CCJ steers the network away from.

As CCJ and QDA prefer different subspaces, their improvements are additive as seen from the rows in Tab 1 and some rows of Tab 2. We also ask, whether the summary statistics from the dataset is needed, for QDA. To test this, we fit the class means and covariances of the QDA on the coreset alone, shrinking each class covariance towards the pooled covariance of the coreset and choosing the shrinkage and the temperature by cross-validation within the coreset. On the 16 CIFAR-100 coresets of Table 1 at IPC-10, this QDA adds $+0.65$ points to `std` and $+0.86$ to $+$CCJ on average, against $+3.91$ and $+4.22$ for the QDA fitted on the full training set, and at IPC-50 it lowers `std` by 0.66 points. The QDA therefore needs the class statistics of the full training set.

E Proofs of Propositions in the Theory

In this section, we present the proofs for all the propositions mentioned in Section 3.

E.1 The linearised model

The linearised model of Section 3.1 is built from a network $f_\theta : \mathbb{R}^D \to \mathbb{R}^C$ with a fixed initialisation θ_0, and is given by

$$J_0(x) = \nabla_\theta f_{\theta_0}(x), \qquad f_u^{\text{lin}}(x) = f_{\theta_0}(x) + J_0(x)u, \qquad \Theta_0(x, x') = J_0(x)J_0(x')^\top \in \mathbb{R}^{C\times C}, \tag{15}$$

where $u = \theta - \theta_0$ is the displacement vector in the parameter space, J_0 is the Jacobian at initialisation and Θ_0 is the empirical tangent kernel. Assumption 1(b) is $\Theta_0(x, x') = \Theta(x, x')I_C$.

On the training set $\{(x_i, y_i)\}_{i=1}^N$, the model is trained by unregularised gradient flow with rate η on the mean cross-entropy,

$$\mathcal{L}(u) = \frac{1}{N}\sum_{i=1}^{N}\Big[-f_u^{lin}(x_i)_{y_i} + \log\sum_{c=1}^{C} e^{f_u^{lin}(x_i)_c}\Big], \qquad \dot{u} = -\eta\nabla_u\mathcal{L}(u). \tag{16}$$

The residuals along the flow are $r_i(\tau) = \text{softmax}(f^{\text{lin}}_{u(\tau)}(x_i)) - e_{y_i}$, where e_c is the class-c one-hot vector and $u(\tau)$ is the parameter displacement at training time τ. The output at training time τ used in Section 3 is $f_\tau := f^{\text{lin}}_{u(\tau)}$.

E.2 Proof of Lemma 1

Lemma 1 (Bayes rule of the planted task) *Under Assumption 1(a), in the whitened coordinates $\varphi_j(x) = u_j^\top(x-\mu)/\sqrt{\lambda_j}$, the Bayes logit of class c is linear, with coefficient $l\,\nu_{hi}^c$ on the high-variance subspace, $tl\,\nu_{lo}^c$ on the low-variance subspace and zero elsewhere.*

Proof. We define $v_c = d_c^{hi} + d_c^{lo}$. Our planted input would be $x = x_0 + v_y$ with $x_0 \sim \mathcal{N}(\mu, \Sigma)$ by Assumption 1(a). $P(x \mid y = c) \sim \mathcal{N}(\mu + v_c, \Sigma)$ would be the conditional distribution of the planted

input. Applying the Bayes' rule, followed by using the log density formula of Gaussian distributions and then absorbing all the common terms across classes to $const(x)$ gives us

$$\log P(y=c \mid x) = \log P(x \mid y=c) - \log P(x) - \log C = \langle x-\mu, \Sigma^{\dagger} v_c \rangle - \tfrac{1}{2} v_c^{\top} \Sigma^{\dagger} v_c + \text{const}(x),$$

The planted vectors lie in the range of Σ by construction, so the pseudo-inverse acts as an inverse on them. The Bayes logit is therefore linear in x up to a class constant, with coefficient vector $\Sigma^{\dagger} v_c$.

The whitened coordinate of x is $\varphi(x) = \Lambda^{-1/2} P^{\top}(x-\mu)$ and P is orthogonal, so

$$\langle x-\mu, \Sigma^{\dagger} v_c \rangle = \left(P \Lambda^{1/2} \varphi\right)^{\top} P \Lambda^{-1} P^{\top} v_c = \varphi^{\top} \Lambda^{-1/2} P^{\top} v_c = \langle \varphi(x), \Lambda^{-1/2} P^{\top} v_c \rangle,$$

Substituting the definition of v_c and Eq. 1

$$\Lambda^{-1/2} P^{\top} v_c = l\, \Lambda_{hi}^{-1/2} \Lambda_{hi}^{1/2} \nu_{hi}^{c} \;\oplus\; tl\, \Lambda_{lo}^{-1/2} \Lambda_{lo}^{1/2} \nu_{lo}^{c} \;\oplus\; 0 = l\, \nu_{hi}^{c} \;\oplus\; tl\, \nu_{lo}^{c} \;\oplus\; 0,$$

which is the claim. The Bayes coefficient on a planted feature are exactly l and tl respectively. We use $\oplus$ to indicate that they live in different subspaces and it has to be seen individually as weights for different subspaces. □

E.3 Proof of Lemma 2

Lemma 2 (Acquisition of an isolated feature) *Under Assumption 1, we plant a single feature of predictivity ρ in a subspace of availability ζ_q, and train by Eq. 16. Then the held-out margin of the correct class against any other, on a fresh background, is initially acquired at the rate*

$$\mathbb{E}\big[\dot{m}_{ac}(0)\big] := \frac{d}{d\tau} \mathbb{E}\big[m_{ac}(\tau)\big]\Big|_{\tau=0} = \frac{\eta}{C}\, \rho^2\, \zeta_q + O(\|d\|^3), \tag{17}$$

the same for every $c \neq a$, where the expectation is over the backgrounds and the class unit vectors.

Proof. In the case of an isolated feature, our training input would be $x_i = x_{0,i} + d_{y_i}$, where $d_c = \rho P_q \Lambda_q^{1/2} \nu_q^c$. We test on a image $x = x_0 + d_a$ on a new background and we are concerned about $m_{ac}(\tau) = (e_a - e_c)^{\top} f_{u(\tau)}^{lin}(x)$

The first step is to define the derivative of $f_{u(\tau)}^{lin}(x)$ w.r.t τ. We start with Eq. 15 and then differentiate giving us $\frac{d}{d\tau} f_{u(\tau)}^{lin}(x) = J_0(x)\dot{u}$. We can calculate $\dot{u} = \frac{d}{d\tau} u$ from Eq. 16, where we can write for the per-example loss

$$\nabla_u \mathcal{L} = \frac{1}{N} \sum_i \Big(\frac{\partial f_u^{lin}(x_i)}{\partial u} \Big)^{\top} \nabla_f \ell_i = \frac{1}{N} \sum_i J_0(x_i)^{\top} r_i \tag{18}$$

Here we have used the fact that $\nabla_f \ell = \text{softmax}(f) - e_y$ which we define as the residual r_i.

Substituting Eq. 18 into $\dot{u} = -\eta \nabla_u \mathcal{L}(u)$ from Eq. 16, we get

$$J_0(x)\dot{u} = -\frac{\eta}{N} \sum_i \underbrace{J_0(x) J_0(x_i)^{\top}}_{=\Theta_0(x,x_i)}\, r_i(\tau), \tag{19}$$

As we assume that the kernel is class diagonal, we have the following

$$\frac{d}{d\tau} f_{u(\tau)}^{\text{lin}}(x) = J_0(x)\, \dot{u} = -\frac{\eta}{N} \sum_{i=1}^{N} \Theta_0(x, x_i)\, r_i(\tau) = -\frac{\eta}{N} \sum_{i=1}^{N} \Theta(x, x_i)\, r_i(\tau), \tag{20}$$

We can proceed with the proof of our proposition using the above equation.

We start by differentiating $m_{ac}(\tau)$ and substituting Eq. 20,

$$\dot{m}_{ac}(\tau) = (e_a - e_c)^{\top} \big(-\frac{\eta}{N} \sum_{i=1}^{N} \Theta(x, x_i)\, r_i(\tau) \big)$$

We can write $-(e_a - e_c)^\top r_i = (\mathbf{1}y_i = a - p_{ia}(\tau)) - (\mathbf{1}y_i = c - p_{ic})$, where $p_{ia}(\tau) := \mathrm{softmax}\big(f^{\mathrm{lin}}u(\tau)(x_i)\big)a$. Substituting it in the above equation gives us

$$\dot{m}_{ac}(\tau) = \frac{\eta}{N}\sum_{i=1}^{N}\Theta(x, x_i)\Big[\big(\mathbf{1}\{y_i = a\} - p_{ia}(\tau)\big) - \big(\mathbf{1}\{y_i = c\} - p_{ic}(\tau)\big)\Big], \tag{21}$$

At $\tau = 0$ the initial logits have a uniform class distribution by Assumption 1(d), so we have

$$\dot{m}_{ac}(0) = \frac{\eta}{N}\sum_{i=1}^{N}\Theta(x, x_i)\big(\mathbf{1}\{y_i = a\} - \mathbf{1}\{y_i = c\}\big). \tag{22}$$

The background of the test input x_0 and each training background image $x_{0,i}$ are independent, so for any training example of a given class, the kernel average is the same. We define the average as

$$\bar{\Theta}(d_a, d_b) := \mathbb{E}_{x,x'}\big[\Theta(x + d_a,\, x' + d_b)\big], \tag{23}$$

where d_a, d_b are the shifts added to the background images.

As there are N/C examples in each class, we have the following as the expectation of Eq. 22 over the pair of backgrounds,

$$\mathbb{E}\big[\dot{m}_{ac}(0)\big] = \frac{\eta}{C}\,\mathbb{E}_{\nu^a,\nu^c}\big[\bar{\Theta}(d_a, d_a) - \bar{\Theta}(d_a, d_c)\big]. \tag{24}$$

where ν^a, ν^c are the class unit vectors in the shifts.

We expand $\bar{\Theta}$ to second order in both shifts,

$$\bar{\Theta}(\delta, \delta') = \bar{\Theta}(0,0) + \nabla_1\bar{\Theta}^\top\delta + \nabla_2\bar{\Theta}^\top\delta' + \tfrac{1}{2}\delta^\top\bar{\Theta}_{11}\delta + \delta^\top\bar{\Theta}_{12}\delta' + \tfrac{1}{2}\delta'^\top\bar{\Theta}_{22}\delta' + O(\|d\|^3), \tag{25}$$

with the derivatives taken at $(0, 0)$.

We denote $G = \mathbb{E}_{x,x'}[\nabla_x\nabla_{x'}\Theta(x, x')] = \bar{\Theta}_{12}$, the subscript denotes the input index of the first derivative and second derivative. For example, $\bar{\Theta}_{11}$ means both derivatives are w.r.t the first input parameter.

Substituting the expansion into Eq. 24, and then subtracting term by term, leaves only the term

$$d_a^\top G\, d_a = \rho^2\left(\Lambda^{1/2}v\right)^\top\big(P^\top GP\big)\big(\Lambda^{1/2}v\big) = \rho^2\,\zeta(\nu_q^a),$$

In the above equation, the final equation is by the definition in Eq. 27. The average of this term across class unit vectors would be $\rho^2\zeta_q$ by Eq. 4. Aside from that term, we will have the expansion remainder $O(\|d\|^3)$ Therefore, collecting the terms together we have

$$\mathbb{E}[\dot{m}_{ac}(0)] = \frac{\eta}{C}\rho^2\zeta_q + O(\|d\|^3)$$

□

E.4 Proof of Proposition 3

Proposition 3 (availability factorises into data and architecture). *Under Assumption 1, the availability of each eigendirection, $\zeta_j := \zeta(u_j)$, factorises as*

$$\zeta_j = \lambda_j\, g_j, \qquad g_j := u_j^\top G\, u_j, \qquad G := \mathbb{E}_{x,x'}\big[\nabla_x\nabla_{x'}\Theta(x, x')\big].$$

Hence $\zeta_q = \frac{1}{K}\sum_{j\in q}\lambda_j g_j$.

Proof. For a unit vector v, we can write $\varphi_v(x) = \sum_j v_j\frac{u_j^\top(x-\mu)}{\sqrt{\lambda_j}}$. We can define $w := P\Lambda^{-1/2}v$, so we have $\varphi_v(x) = w^\top(x - \mu)$ and $\mathrm{Var}(\varphi_v) = w^\top\Sigma w = \|v\|^2 = 1$.

We then express the availability in terms of w, by substituting the above value of $\varphi_v(x)$ into Eq. 3, giving us

$$\zeta(v) = \mathbb{E}_{x,x'}\left[\Theta(x,x')\, w^\top (x-\mu)(x'-\mu)^\top w\right] = w^\top M w, \\ M := \mathbb{E}_{x,x'}\left[\Theta(x,x')(x-\mu)(x'-\mu)^\top\right]. \tag{26}$$

The backgrounds are Gaussian by Assumption 1(a), and Stein's identity for $x \sim \mathcal{N}(\mu, \Sigma)$ states that $\mathbb{E}[(x-\mu)h(x)] = \Sigma\, \mathbb{E}[\nabla h(x)]$ for differentiable h with integrable derivative. Apply it in the first argument with $h(\cdot) = \Theta(\cdot, x')$ at fixed x', then in the second argument, because the two draws are independent and the derivatives are integrable by Assumption 1(c):

$$M = \Sigma\, \mathbb{E}_{x,x'}\left[\nabla_x \Theta(x,x')\,(x'-\mu)^\top\right] = \Sigma\, \mathbb{E}_{x,x'}\left[\nabla_x \nabla_{x'} \Theta(x,x')\right] \Sigma = \Sigma G \Sigma.$$

Substituting into Eq. 26 and using $\Sigma w = P\Lambda P^\top P \Lambda^{-1/2} v = P\Lambda^{1/2} v$,

$$\zeta(v) = \big(\Sigma w\big)^\top G \big(\Sigma w\big) = \big(P\Lambda^{1/2} v\big)^\top G \big(P\Lambda^{1/2} v\big) = \big(\Lambda^{1/2} v\big)^\top \big(P^\top G P\big) \big(\Lambda^{1/2} v\big). \tag{27}$$

Taking $v = e_j$, the direction of u_j, only the diagonal entry of $P^\top G P$ enters, and $\zeta(u_j) = \lambda_j\, u_j^\top G u_j = \lambda_j g_j$, which is Eq. 9. Averaging over $j \in q$ with Eq. 4 gives $\zeta_q = \frac{1}{K} \sum_{j \in q} \lambda_j g_j$. □

E.5 Proof of Proposition 1

Proposition 1 (preference along training). *Under Assumptions 1 and 2, for every training step $\tau \geq 0$ the preference changes sign at*

$$t_\star(\tau)^2 \approx \frac{L_{hi}(\tau)}{L_{lo}(\tau)}.$$

In particular, $t_\star(\tau)^2 \to \zeta_{hi}/\zeta_{lo}$ as $\tau \to 0$.

Proof. In the case of a conflict, our test input would be $x = x_0 + d_a^{hi} + d_b^{lo}$, where x_0 is a new background image, independent of the training set, and we are concerned about $m_{ab}(\tau) = (e_a - e_b)^\top f^{\text{lin}}_{u(\tau)}(x)$.

We write the margin as an integral over training time. We differentiate $m_{ab}(\tau)$ and substitute Eq. 20, exactly as in the proof of Lemma 2 with the class c replaced by b, which gives us

$$\dot{m}_{ab}(\tau) = \frac{\eta}{N} \sum_{i=1}^{N} \Theta(x, x_i)\, s_i(\tau), \qquad s_i := \big(\mathbf{1}\{y_i = a\} - p_{ia}\big) - \big(\mathbf{1}\{y_i = b\} - p_{ib}\big). \tag{28}$$

The initial logits are equal across classes by Assumption 1(d), so $m_{ab}(0) = 0$, and integrating Eq. 28 from 0 to τ gives us

$$m_{ab}(\tau) = \frac{\eta}{N} \sum_{i=1}^{N} \int_0^\tau \Theta(x, x_i)\, s_i(s)\, ds. \tag{29}$$

We would like to find the average of the kernel across all the backgrounds so we make use of Mercer's theorem. Mercer's theorem states that any continuous, symmetric and positive semi-definite kernel can be expressed as a convergent sum of products of its eigenfunctions, weighted by its eigenvalues. We take the eigenfunctions with respect to the background distribution. By Assumption 2, the whitened coordinates φ_j are among them, with eigenvalues ζ_j, and we write $\zeta_0 := \mathbb{E}_{x,x'}[\Theta(x,x')]$ for the mean of the kernel, so

$$\Theta(x,x') = \zeta_0 + \sum_{j=1}^{D} \zeta_j\, \varphi_j(x) \varphi_j(x') + \Theta^\perp(x,x'), \tag{30}$$

where $\Theta^\perp$ collects the rest. Since the φ_j are centred eigenfunctions, $\Theta^\perp$ is orthogonal in each argument to every φ_j, that is, $\mathbb{E}_{x'}[\Theta^\perp(x,x')\varphi_j(x')] = 0$ for every x, with x' an independent background, and its mean over two independent backgrounds is zero.

Next, we need the whitened coordinates of a planted input. The whitened coordinates are affine in x, and $u_j^\top d_c^q = \rho_q \sqrt{\lambda_j}\, \nu_{q,j}^c$ for $j \in q$ and zero for $j \notin q$ by Eq. 1, so planting moves the whitened coordinates of block q by exactly $\rho_q \nu_q^c$,

$$\varphi_q(x_i) = \rho_q\, \nu_q^{y_i} + z_{q,i}, \qquad z_{q,i} := \varphi_q(x_{0,i}). \tag{31}$$

The same holds for the conflict input. As $\mathbb{E}[\varphi_j(x_0)] = 0$, we have $\mathbb{E}_{x_0}[\varphi_j(x)] = \rho_{hi}\nu_{hi,j}^a$ for $j \in hi$, $\rho_{lo}\nu_{lo,j}^b$ for $j \in lo$ and 0 otherwise.

Substituting this into the average of Eq. 30 over x_0 gives us

$$\begin{aligned} \mathbb{E}_{x_0}\big[\Theta(x, x_i)\big] &= \zeta_0 + \rho_{hi} \sum_{j \in hi} \zeta_j \nu_{hi,j}^a \varphi_j(x_i) + \rho_{lo} \sum_{j \in lo} \zeta_j \nu_{lo,j}^b \varphi_j(x_i) + R(x_i), \\ R(x_i) &:= \mathbb{E}_{x_0}\big[\Theta^\perp(x, x_i)\big]. \end{aligned} \tag{32}$$

When we substitute Eq. 32 into Eq. 29, we get three terms

The constant. The first term is $\zeta_0 \frac{1}{N}\sum_i s_i$, and it has zero expectation.

The linear modes. We start from the definition in Eq. 5 with $(c, c') = (a, b)$. We write $f_{u(\tau)}^{\mathrm{lin}}(x_0)$ as its initial value plus the integral of Eq. 20, where the initial value drops out of the difference $(e_a - e_b)^\top$ by Assumption 1(d). Taking the expectation over x_0 and applying Assumption 2 at the training inputs gives us

$$\rho_{hi} L_{hi}(\tau) = \mathbb{E}_{x_0}\Big[(e_a - e_b)^\top f_{u(\tau)}^{\mathrm{lin}}(x_0)\, \varphi_{\nu_{hi}^a}(x_0)\Big] = \frac{\eta}{N} \sum_i \int_0^\tau \sum_{j \in hi} \zeta_j\, \nu_{hi,j}^a\, \varphi_j(x_i)\, s_i(s)\, ds, \tag{33}$$

which is exactly the high-variance subspace term of Eq. 32 integrated against s_i. The low-variance subspace term is the same computation with $(c, c') = (b, a)$ and direction ν_{lo}^b. Since $(e_a - e_b) = -(e_b - e_a)$, it enters as $-\rho_{lo} \cdot \rho_{lo} L_{lo}(\tau)$. Here we have not taken an average over the class unit vectors, and we have not assumed anything about the residuals. Whatever correlation training builds between $p_{ic}(s)$ and the class unit vectors is carried inside L_q.

The remainder. We write $\delta := d_a^{hi} + d_b^{lo}$ and expand $R(x_i)$ in δ about the unplanted probe.

- At zeroth order, the term is $\frac{1}{N}\sum_i \mathbb{E}_{x_0}[\Theta^\perp(x_0, x_i)]\, s_i(s)$, and it has zero expectation.
- At first order, $\mathbb{E}_{x_0}[\nabla_x \Theta^\perp(x_0, x_i)] = 0$. By Stein's identity for the Gaussian background of Assumption 1(a), $\mathbb{E}[\nabla\psi(x_0)] = \Sigma^{-1}\mathbb{E}[(x_0 - \mu)\psi(x_0)]$ with $\psi = \Theta^\perp(\cdot, x_i)$, and the right side vanishes because $x_0 - \mu = P\Lambda^{1/2}\varphi(x_0)$ and ψ is orthogonal to every whitened coordinate.
- At second order, the term is $\delta^\top H(x_i)\delta$ with $H(x_i) := \mathbb{E}_{x_0}[\nabla_x \nabla_x^\top \Theta^\perp(x_0, x_i)]$, and it multiplies $\frac{1}{N}\sum_i s_i(s)$.

For the second-order term, we separate what depends on the class unit vectors from what does not. Let s_i^0 be s_i computed along the flow trained on the unplanted inputs $x_{0,i}$ with the same labels. The flow is smooth in the planting scale, so $s_i = s_i^0 + O(\|d\|)$ and $H(x_i) = H(x_{0,i}) + O(\|d\|)$. As $\delta\delta^\top$ is already $O(\|d\|^2)$, both corrections contribute $O(\|d\|^3)$, which leaves

$$\frac{1}{N}\sum_{i=1}^N \delta^\top H(x_{0,i})\, \delta\, s_i^0(s).$$

Here δ depends only on the class unit vectors, and $H(x_{0,i})$ and s_i^0 only on the backgrounds and labels, so averaging over the class unit vectors replaces $\delta\delta^\top$ by the fixed matrix $\mathbb{E}[\delta\delta^\top]$. Exchanging the labels $a \leftrightarrow b$ in the training set preserves their joint law and leaves the backgrounds fixed. Because the initial softmax is uniform (Assumption 1(d)), it also exchanges p_{ia} and p_{ib} along the flow, so $s_i^0 \mapsto -s_i^0$ and this term has zero expectation. The remainder is therefore $O(\|d\|^3)$, uniformly in τ on compact sets.

Therefore, collecting the three terms together gives us, for every $\tau \geq 0$,

$$\mathbb{E}\big[m_{ab}(\tau)\big] = (\rho_{hi}^2 L_{hi}(\tau) - \rho_{lo}^2 L_{lo}(\tau)) + O(\|d\|^3). \tag{34}$$

Dropping the remainder, the preference changes sign where $\rho_{hi}^2 L_{hi}(\tau) - \rho_{lo}^2 L_{lo}(\tau)$ does, which for $L_{lo}(\tau) > 0$ is Eq. 7.

The initial rate. At $\tau = 0$ the softmax is uniform by Assumption 1(d), so $s_i(0) = \mathbf{1}\{y_i = a\} - \mathbf{1}\{y_i = b\}$ depends only on the label. We differentiate Eq. 33 at $\tau = 0$ and take the expectation over the training backgrounds, for which $\mathbb{E}[\varphi_j(x_i)] = \rho_{hi}\nu_{hi,j}^{y_i}$ for $j \in hi$ by Eq. 31. As each class holds a fraction $1/C$ of the training set, we have

$$\rho_{hi}\dot{L}_{hi}(0) = \frac{\eta}{N}\sum_i \sum_{j\in hi} \zeta_j\, \nu_{hi,j}^a\, \rho_{hi}\nu_{hi,j}^{y_i}\big(\mathbf{1}\{y_i = a\} - \mathbf{1}\{y_i = b\}\big) = \frac{\eta\rho_{hi}}{C}\sum_{j\in hi} \zeta_j\, \nu_{hi,j}^a\big(\nu_{hi,j}^a - \nu_{hi,j}^b\big). \tag{35}$$

The class unit vectors ν_q^c are independent across classes and across blocks, and uniform on the unit sphere of $\mathbb{R}^K$, so

$$\mathbb{E}\big[\nu_{q,j}^c\big] = 0, \qquad \mathbb{E}\big[\nu_{q,j}^c \nu_{q,k}^c\big] = \frac{\delta_{jk}}{K}, \qquad \mathbb{E}\big[\langle \nu_q^c, \nu_q^{c'}\rangle\big] = \mathbf{1}\{c = c'\}. \tag{36}$$

Averaging Eq. 35 over the class unit vectors with Eq. 36, we have $\mathbb{E}[(\nu_{hi,j}^a)^2] = 1/K$ and $\mathbb{E}[\nu_{hi,j}^a \nu_{hi,j}^b] = 0$, so $\dot{L}_{hi}(0) = \eta\zeta_{hi}/C$ by Eq. 4, at any N. The low block is the same computation. The low-variance shifts have no component along the high-variance whitened coordinates, and vice versa, so the two planted features do not interact in this rate. Since $L_q(0) = 0$, we have $t_\star(\tau)^2 = L_{hi}(\tau)/L_{lo}(\tau) \to \dot{L}_{hi}(0)/\dot{L}_{lo}(0) = \zeta_{hi}/\zeta_{lo}$ as $\tau \to 0$. □

E.6 Proof of Proposition 2

Proposition 2 (recovery in the population limit). *Under Assumptions 1 and 2, in the population limit $N \to \infty$*

$$L_q(\tau) \approx \frac{1}{K}\sum_{j\in q}\big(1 - e^{-\eta\zeta_j\tau/C}\big).$$

If every high-variance direction is more available than every low-variance one ($\zeta_j > \zeta_k$ for all $j \in hi$, $k \in lo$), then $1 \le t_\star(\tau)^2 \le \zeta_{hi}/\zeta_{lo}$ for every $\tau > 0$, and $t_\star(\tau)$ decreases strictly to 1 as $\tau \to \infty$.

Proof. **The population form.** We let $N \to \infty$ at fixed class balance, so that the empirical average in Eq. 20 becomes an expectation over (Y, X) with Y uniform and $X = x_0 + d_Y$. We scale every planted shift by ϵ and write $f_c^\epsilon(x, \tau)$ for the resulting flow.

The first step is to look at the flow without any planted features. At $\epsilon = 0$ the inputs are independent of the label, the residual $\mathbb{E}[\mathbf{1}\{Y = c\} - 1/C]$ vanishes and the logits stay equal across classes, so $p_c \equiv 1/C$.

We then differentiate the population flow in ϵ at 0 and write $F_c := \partial_\epsilon f_c^\epsilon|_{\epsilon=0}$. For this we need two derivatives. By Assumption 1(a), the class-conditional density has the Gaussian expansion $dZ_{\epsilon,c}/dZ_0(x) = 1 + \epsilon\, b_c(x) + O(\epsilon^2)$, where by Eq. 31

$$b_c(x) = d_c^\top \Sigma^{-1}(x - \mu) = \rho_{hi}\,\varphi_{\nu_{hi}^c}(x) + \rho_{lo}\,\varphi_{\nu_{lo}^c}(x). \tag{37}$$

The softmax at the uniform vector has derivative $\partial_\epsilon p_c = (F_c - \bar{F})/C$, with $\bar{F} := \frac{1}{C}\sum_k F_k$. We write $\mathcal{T}$ for the kernel's integral operator over the backgrounds, $(\mathcal{T}g)(x) := \mathbb{E}_{x'}[\Theta(x, x')g(x')]$ with x' an independent background, so that Assumption 2 reads $\mathcal{T}\varphi_j = \zeta_j\varphi_j$. Substituting both derivatives into the population flow gives us

$$\dot{F}_c = \frac{\eta}{C}\,\mathcal{T}\big[(b_c - \bar{b}) - (F_c - \bar{F})\big], \qquad F_c(\cdot, 0) = 0, \tag{38}$$

where $\bar{b} := \frac{1}{C}\sum_k b_k$. Summing Eq. 38 over c shows $\bar{F} \equiv 0$.

Each $b_c - \bar{b}$ lies in the span of the whitened coordinates, which $\mathcal{T}$ maps to itself since $\mathcal{T}\varphi_j = \zeta_j\varphi_j$. So F_c stays in that span, and its coefficients obey the decoupled scalar equations $\dot{\alpha}_{c,j} = \frac{\eta\zeta_j}{C}(\beta_{c,j} -$

$\alpha_{c,j}$), with $\beta_{c,j} = \rho_q(\nu^c_{q,j} - \bar\nu_{q,j})$ for $j \in q$ and $\bar\nu_q := \frac{1}{C}\sum_k \nu^k_q$. Solving them gives us

$$F_c(x,\tau) = \sum_q \rho_q \sum_{j\in q} \big(\nu^c_{q,j} - \bar\nu_{q,j}\big)\big(1 - e^{-\eta\zeta_j\tau/C}\big)\,\varphi_j(x). \tag{39}$$

We insert $f^{\mathrm{lin}}_{u(\tau)} = f^{\mathrm{lin}}_{u(0)} + \epsilon F + O(\epsilon^2)$ into Eq. 5, absorbing ϵ into the predictivities, and use $\mathbb{E}[\varphi_j(x_0)\varphi_k(x_0)] = \delta_{jk}$, which gives us

$$L_q(\tau) = \sum_{j\in q}\big(1 - e^{-\eta\zeta_j\tau/C}\big)\,\nu^c_{q,j}\big(\nu^c_{q,j} - \nu^{c'}_{q,j}\big) + O(\|d\|). \tag{40}$$

Here the $\bar\nu_q$ cancel in the difference of the two classes. Averaging Eq. 40 over the class unit vectors with Eq. 36 gives us Eq. 8. This average is legitimate: the recovery factors $1 - e^{-\eta\zeta_j\tau/C}$ do not depend on the class unit vectors, because at $\epsilon = 0$ the population flow does not move and the linear response is driven by an operator independent of them.

Consequences. Expanding Eq. 8 at small τ gives us $L_q(\tau) = \eta\zeta_q\tau/C + O(\tau^2)$ by Eq. 4, consistent with the limit in Proposition 1. Differentiating Eq. 34 at $\tau = 0$ gives us $\frac{\eta}{C}(\rho^2_{hi}\zeta_{hi} - \rho^2_{lo}\zeta_{lo})$, which is Eq. 17 applied to each planted feature in turn.

For the bracket, we write $1 - e^{-\eta\zeta\tau/C} = \zeta\, g_\tau(\zeta)$ with $g_\tau(\zeta) := (1 - e^{-\eta\zeta\tau/C})/\zeta$, which is positive and strictly decreasing in ζ for $\tau > 0$. If $\zeta_j \geq \zeta_k$ for all $j \in hi$ and $k \in lo$, every summand of L_{hi} is at least every summand of L_{lo}, so $L_{hi} \geq L_{lo}$. For the upper bound, we have

$$L_{hi}\,\zeta_{lo} = \frac{1}{K^2}\sum_{j\in hi}\sum_{k\in lo}\zeta_j\zeta_k\, g_\tau(\zeta_j) \leq \frac{1}{K^2}\sum_{j\in hi}\sum_{k\in lo}\zeta_j\zeta_k\, g_\tau(\zeta_k) = \zeta_{hi}\,L_{lo}, \tag{41}$$

so $1 \leq L_{hi}/L_{lo} \leq \zeta_{hi}/\zeta_{lo}$. As $\tau \to \infty$ every summand tends to 1, so $t_\star \to 1$. For the monotonicity, Corollary 1 applies to the fractions of Eq. 8 for any separated spectrum, so $t_\star(\tau)$ is strictly decreasing in τ. □

F Both Claims Under Squared Loss

Under cross-entropy the residuals $r_i(\tau)$ correlate with the class unit vectors as soon as training starts, and that correlation is what confines Lemma 2 to $\tau = 0$ and forces the population limit in Proposition 2. We don't have this restriction in the case of squared loss. We still perform the analysis in the same planted task from Section 3.1

F.1 Kernel ridge regression on the planted task

We keep the linearised model Eq. 15 and the class-diagonal kernel of Assumption 1(b), and we drop the orthogonal part of the Mercer decomposition Eq. 30.

Assumption 3 (diagonal kernel) $\Theta(x,x') = \zeta_0 + \sum_{j=1}^{D}\zeta_j\varphi_j(x)\varphi_j(x')$ *with* $\zeta_j > 0$*, that is,* $\Theta^\perp \equiv 0$ *in Eq. 30.*

This is Assumption 2 with its remainder term set to zero. Under it the tangent model is a linear model in the features

$$\psi_j(x) := \sqrt{\zeta_j}\,\varphi_j(x), \qquad \langle\psi(x),\psi(x')\rangle = \Theta(x,x') - \zeta_0, \tag{42}$$

so training the tangent model is ridge regression in ψ. We regress the centred one-hot targets $Y_c := \mathbf{1}\{y = c\} - 1/C$ under

$$\mathcal{L}_{\mathrm{sq}}(w) = \tfrac{1}{2}\,\mathbb{E}\big[\big(w_c^\top\psi(X) - Y_c\big)^2\big] + \tfrac{\lambda}{2}\,\|w_c\|^2, \tag{43}$$

with $\lambda \geq 0$ and $f_c(x) = w_c^\top\psi(x)$. Ridge stands in for early stopping: the $\lambda \to 0$ limit is the interpolating solution and large λ is the start of training. Section F.6 explains why the two act alike, and Proposition 5 treats training time directly, by gradient flow.

Squared loss regresses probabilities rather than logits, so the population optimum of Eq. 43 at $\lambda = 0$ is the Bayes coefficient of Lemma 1 times the softmax link factor $1/C$. We therefore normalise the recovered fraction of Eq. 5 by that optimum rather than by the Bayes logit weight. The two differ by the constant $1/C$, which cancels in $t_\star$. The remainders below are $O(\|d\|^2)$ relative rather than $O(\|d\|)$, because no expansion of the kernel in the planted shifts is needed once $\Theta^\perp$ is gone.

Lemma 3 (threshold of a linear fit) *Let $f_c = \sum_j \beta_{c,j}\varphi_j$ for every class c, with coefficients independent of the test background, and let L_{hi} and L_{lo} be Eq. 5, normalised as above, for the pairs $(c, c') = (a, b)$ and (b, a). Then on the conflict input x_{ab},*

$$C\,\mathbb{E}_{x_0}\big[m_{ab}\big] = \rho_{hi}^2 L_{hi} - \rho_{lo}^2 L_{lo} = l^2\big(L_{hi} - t^2 L_{lo}\big), \tag{44}$$

so if the L_q do not depend on t the preference changes sign at $t_\star^2 = L_{hi}/L_{lo}$. The same holds after averaging over training sets and class unit vectors.

Proof. Since $\mathbb{E}_{x_0}[\varphi_j\varphi_k] = \delta_{jk}$, Eq. 5 reduces to

$$\frac{\rho_q}{C}\,L_q = \sum_{j\in q}\big(\beta_{c,j} - \beta_{c',j}\big)\nu_{q,j}^c. \tag{45}$$

The whitened coordinates are affine, so by Eq. 31 the conflict input has $\varphi_j(x_{ab}) = \varphi_j(x_0) + \rho_{hi}\nu_{hi,j}^a$ for $j \in hi$ and $\varphi_j(x_0) + \rho_{lo}\nu_{lo,j}^b$ for $j \in lo$. As $\mathbb{E}_{x_0}[\varphi_j(x_0)] = 0$,

$$\mathbb{E}_{x_0}\big[m_{ab}\big] = \rho_{hi}\sum_{j\in hi}(\beta_{a,j} - \beta_{b,j})\nu_{hi,j}^a + \rho_{lo}\sum_{j\in lo}(\beta_{a,j} - \beta_{b,j})\nu_{lo,j}^b,$$

which is Eq. 44 by Eq. 45, with $\rho_{hi} = l$ and $\rho_{lo} = tl$. Averaging is linear, so it preserves the identity. □

F.2 The recovered fraction under ridge

Lemma 4 (ridge recovered fraction) *Under Assumption 3, Assumption 1(a) and the planted task of Section 3.1, the population ridge solution of Eq. 43 recovers the coefficient on φ_j with the factor $\zeta_j/(\zeta_j + \lambda)$, and, in expectation over the backgrounds and the class unit vectors,*

$$L_q(\lambda) = \frac{1}{K}\sum_{j\in q}\frac{\zeta_j}{\zeta_j + \lambda} \;+\; O(\|d\|^2). \tag{46}$$

Proof. Setting the gradient of Eq. 43 with respect to w_c to zero gives the population ridge solution

$$w_c = \big(Q + \lambda I\big)^{-1} b_c, \qquad Q := \mathbb{E}[\psi\psi^\top], \qquad b_c := \mathbb{E}[\psi\, Y_c]. \tag{47}$$

By Eq. 31, $\varphi_j(X) = \rho_q \nu_{q,j}^Y + z_j$ for $j \in q$, where $z = \varphi(X^0)$ is centred, whitened and independent of Y. The cross terms therefore vanish, and with $\psi_j = \sqrt{\zeta_j}\,\varphi_j$ we have, for $j \in q$,

$$Q = \mathrm{diag}(\zeta) + O(\|d\|^2), \qquad b_{c,j} = \frac{\sqrt{\zeta_j}\,\rho_q}{C}\big(\nu_{q,j}^c - \bar\nu_{q,j}\big), \qquad \bar\nu_q := \frac{1}{C}\sum_k \nu_q^k, \tag{48}$$

and $b_{c,j} = 0$ outside the two blocks; the $O(\|d\|^2)$ term of Q comes from the planted shifts, and b_c is the driving term of Eq. 38. That term multiplies a signal of order $\|d\|$, so each direction is solved on its own up to $O(\|d\|^3)$, and the coefficient of f_c on φ_j is

$$\beta_{c,j} := \sqrt{\zeta_j}\,w_{c,j} = \frac{\zeta_j}{\zeta_j + \lambda}\cdot\frac{\rho_q(\nu_{q,j}^c - \bar\nu_{q,j})}{C} + O(\|d\|^3). \tag{49}$$

At $\lambda = 0$ this is the population optimum, so the recovery factor is $\zeta_j/(\zeta_j + \lambda)$, the $N \to \infty$ case of the learnability of Simon et al. (2023). Inserting Eq. 49 into Eq. 45, the $\bar\nu_q$ cancels and

$$L_q(\lambda) = \sum_{j\in q}\frac{\zeta_j}{\zeta_j + \lambda}\,\nu_{q,j}^c\big(\nu_{q,j}^c - \nu_{q,j}^{c'}\big) + O(\|d\|^2). \tag{50}$$

The factors depend on the kernel alone, so averaging over the class unit vectors with Eq. 36, $\mathbb{E}[(\nu_{q,j}^c)^2] = 1/K$ and $\mathbb{E}[\nu_{q,j}^c\nu_{q,j}^{c'}] = 0$, gives Eq. 46. □

Eq. 46 is Eq. 8 with $1 - e^{-\eta\zeta_j\tau/C}$ replaced by $\zeta_j/(\zeta_j + \lambda)$. Both are per-direction saturating functions of that direction's availability, rising from 0 to 1.

F.3 MONOTONICITY LEMMA

Both recovered fractions have the form of a block average of a per-direction function of availability, so we prove the bracket and the monotonicity once, for a general such function S.

Lemma 5 (bracket and monotonicity) *Let $g : (0,\infty) \to (0,\infty)$ be strictly decreasing, $S(\zeta) := \zeta\, g(\zeta)$ and*

$$L_q := \frac{1}{K}\sum_{j\in q} S(\zeta_j), \qquad \zeta_q := \frac{1}{K}\sum_{j\in q}\zeta_j.$$

Suppose S is non-decreasing and the blocks are separated, $\zeta_j > \zeta_k$ for every $j \in hi$ and $k \in lo$. Then

$$1 \;\le\; \frac{L_{hi}}{L_{lo}} \;\le\; \frac{\zeta_{hi}}{\zeta_{lo}}, \tag{51}$$

and the first inequality is strict when S is strictly increasing. If moreover $S = S_\theta$ depends differentiably on a parameter θ and $\omega_\theta(\zeta) := \partial_\theta \log S_\theta(\zeta)$ is strictly monotone in ζ, then L_{hi}/L_{lo} is strictly monotone in θ, strictly decreasing when ω_θ is decreasing in ζ and strictly increasing when ω_θ is increasing.

Proof. For Eq. 51, S non-decreasing and the separation give $S(\zeta_j) \ge S(\zeta_k)$ termwise, strictly if S is strictly increasing, hence $L_{hi} \ge L_{lo}$. For the upper bound, g decreasing and the separation give $g(\zeta_j) \le g(\zeta_k)$, so

$$L_{hi}\,\zeta_{lo} = \frac{1}{K^2}\sum_{j\in hi}\sum_{k\in lo}\zeta_j\zeta_k\, g(\zeta_j) \le \frac{1}{K^2}\sum_{j\in hi}\sum_{k\in lo}\zeta_j\zeta_k\, g(\zeta_k) = \zeta_{hi}\, L_{lo}.$$

For the monotonicity, write $\pi_{q,j} := S_\theta(\zeta_j)/\sum_{k\in q} S_\theta(\zeta_k)$, a probability weight on block q. Then

$$\partial_\theta \log L_q = \frac{\sum_{j\in q}\partial_\theta S_\theta(\zeta_j)}{\sum_{j\in q} S_\theta(\zeta_j)} = \sum_{j\in q}\pi_{q,j}\,\omega_\theta(\zeta_j),$$

a weighted average of ω_θ over the availabilities of block q. Because the blocks are separated, for ω_θ strictly decreasing

$$\partial_\theta \log L_{hi} \;\le\; \max_{j\in hi}\omega_\theta(\zeta_j) \;<\; \min_{k\in lo}\omega_\theta(\zeta_k) \;\le\; \partial_\theta \log L_{lo}$$

whatever the weights are, and therefore $\partial_\theta \log(L_{hi}/L_{lo}) < 0$. The increasing case is the same computation with the inequalities reversed. □

The lemma applies to both time and ridge.

Corollary 1 (Early Stopping and Explicit Ridge) *Take $\theta = \tau$ and $S_\tau(\zeta) = 1 - e^{-\eta\zeta\tau/C}$, so that $g_\tau(\zeta) = (1 - e^{-\eta\zeta\tau/C})/\zeta$. Then g_τ is strictly decreasing, S_τ is increasing, and $\omega_\tau(\zeta) = \frac{\eta}{C}\,\zeta/(e^{\eta\zeta\tau/C} - 1)$ is strictly decreasing in ζ; Lemma 5 therefore gives Eq. 51 for the recovered fractions of Eq. 8 and makes $t_\star(\tau)$ strictly decreasing in τ. Take instead $\theta = \lambda$ and $S_\lambda(\zeta) = \zeta/(\zeta + \lambda)$, so that $g_\lambda(\zeta) = 1/(\zeta + \lambda)$. Then g_λ is strictly decreasing, S_λ is increasing, and $\omega_\lambda(\zeta) = -1/(\zeta + \lambda)$ is strictly increasing in ζ; Lemma 5 gives Eq. 51 for the fractions of Eq. 46 and makes $t_\star(\lambda)$ strictly increasing in λ.*

Proof. With $u := \eta\zeta\tau/C$, $g_\tau = \frac{\eta\tau}{C}\,(1 - e^{-u})/u$ and $\omega_\tau = \frac{1}{\tau}\,u/(e^u - 1)$; both functions of u are strictly decreasing on $u > 0$, as the numerators of their derivatives, $(1+u)e^{-u} - 1$ and $e^u - 1 - ue^u$, vanish at 0 and decrease. The ridge case is immediate. In both cases $t_\star^2 = L_{hi}/L_{lo}$, by Proposition 1 and by Lemma 3 respectively. □

F.4 CLAIM 1

Proposition 4 (Claim 1 under squared loss) *Under Assumption 3, Assumption 1(a) and $\lambda > 0$, suppose the availabilities are separated across the blocks, $\zeta_j > \zeta_k$ for every $j \in hi$ and $k \in lo$. Then for every $\lambda > 0$*

$$1 \;<\; t_\star(\lambda)^2 = \frac{L_{hi}(\lambda)}{L_{lo}(\lambda)} \;\le\; \frac{\zeta_{hi}}{\zeta_{lo}}, \tag{52}$$

with $t_\star(\lambda)^2 \to \zeta_{hi}/\zeta_{lo}$ as $\lambda \to \infty$, $t_\star(\lambda) \to 1$ as $\lambda \to 0$, and $t_\star$ strictly increasing in λ. If in addition the hypotheses of Proposition 3 hold and the kernel gains g_j do not reverse the ordering of the variances, the separation hypothesis is automatic, since $\zeta_j = \lambda_j g_j$ and the high block is taken above the low one in the spectrum of Σ.

Proof. At leading order in $\|d\|$, Lemma 4 gives $L_q(\lambda) = \frac{1}{K}\sum_{j\in q} S_\lambda(\zeta_j)$ with $S_\lambda(\zeta) = \zeta/(\zeta+\lambda)$, which does not depend on t, so Lemma 3 gives $t_\star^2 = L_{hi}/L_{lo}$. Corollary 1 gives the bracket and the monotonicity, and the lower bound is strict because S_λ is strictly increasing. As $\lambda \to 0$, $S_\lambda(\zeta) \to 1$ for every $\zeta > 0$, so both fractions tend to 1; as $\lambda \to \infty$, $L_q(\lambda) = \zeta_q/\lambda + O(\lambda^{-2})$, so the ratio tends to ζ_{hi}/ζ_{lo}. For the last sentence, $\zeta_j = \lambda_j g_j$ by Proposition 3 (with λ_j the eigenvalue of Σ), so the separation is $\lambda_j g_j > \lambda_k g_k$. □

A feature in the low-variance block is exploited only if it is strictly more predictive than its high-variance competitor, and the advantage it needs never exceeds $\sqrt{\zeta_{hi}/\zeta_{lo}}$.

The ridge λ stands in for the length of training. Unregularised gradient flow can be treated directly, in the population limit, and gives the same conclusion at every training time.

Proposition 5 (Claim 1 along training under squared loss) *Under Assumption 3 and Assumption 1(a), train Eq. 43 with $\lambda = 0$ by population gradient flow $\dot{w}_c = -\eta \nabla_{w_c} \mathcal{L}_{\mathrm{sq}}(w)$ from $w_c(0) = 0$. Then, in expectation over the backgrounds and the class unit vectors,*

$$L_q(\tau) = \frac{1}{K}\sum_{j\in q}\big(1 - e^{-\eta\zeta_j\tau}\big) + O(\|d\|^2). \tag{53}$$

If the availabilities are separated across the blocks, $\zeta_j > \zeta_k$ for every $j \in hi$ and $k \in lo$, then for every $\tau > 0$

$$1 \;<\; t_\star(\tau)^2 \;\le\; \frac{\zeta_{hi}}{\zeta_{lo}},$$

with $t_\star(\tau)^2 \to \zeta_{hi}/\zeta_{lo}$ as $\tau \to 0$, $t_\star(\tau) \to 1$ as $\tau \to \infty$, and $t_\star$ strictly decreasing in τ.

Proof. With $\lambda = 0$ the flow is $\dot{w}_c = -\eta(Qw_c - b_c)$, in the notation of Eq. 47. As $Q = \mathrm{diag}(\zeta) + O(\|d\|^2)$ by Eq. 48, each direction relaxes on its own, $w_{c,j}(\tau) = \big(1 - e^{-\eta\zeta_j\tau}\big) b_{c,j}/\zeta_j$, up to $O(\|d\|^3)$ at each τ. The recovery factor is therefore $1 - e^{-\eta\zeta_j\tau}$, and the last two steps of the proof of Lemma 4 give Eq. 53. The rest is the proof of Proposition 4, with the first half of Corollary 1 and η/C replaced by η. As $\tau \to 0$, $L_q(\tau) = \eta\zeta_q\tau + O(\tau^2)$ and the ratio tends to ζ_{hi}/ζ_{lo}; as $\tau \to \infty$ every factor tends to 1. □

Together with Proposition 4, Claim 1 holds under squared loss at every ridge and at every training time of unregularised gradient flow, in the population limit.

F.5 CLAIM 2

Claim 2 is about the budget, and the population objective Eq. 43 has no n in it. At finite sample size the design matrix replaces $\mathrm{diag}(\zeta)$ by its empirical counterpart, and the standard description of that replacement is that the estimator behaves like population ridge at an inflated level of regularisation.

Assumption 4 (effective ridge) *Let $\mathrm{df}(\mu) := \sum_{k=1}^{D} \zeta_k/(\zeta_k + \mu)$. In the proportional regime $D, N \to \infty$ with $D/N \to \gamma \in (0, \infty)$ and the spectrum $\{\zeta_k\}$ converging, the empirical ridge solution of Eq. 43 on $N = Cn$ samples recovers the coefficient on φ_j with the factor $\zeta_j/(\zeta_j + \lambda_\star) + o(1)$, where $\lambda_\star > 0$ solves*

$$\lambda_\star = \lambda + \frac{\lambda_\star}{N}\,\mathrm{df}(\lambda_\star). \tag{54}$$

Eq. 54 is the usual self-consistent equation for the effective regularisation of ridge regression with anisotropic Gaussian design (Bordelon et al., 2020; Canatar et al., 2021; Jacot et al., 2020; Simon et al., 2023).

Lemma 6 (the effective ridge falls with the budget) *Fix $\lambda \geq 0$ and a spectrum with $\zeta_k > 0$. If $\lambda > 0$, Eq. 54 has a unique root $\lambda_\star(N) > \lambda$ for every N; if $\lambda = 0$, it has a unique root $\lambda_\star(N) > 0$ exactly when $N < D$. In both cases $\lambda_\star$ is strictly decreasing in N, with $\lambda_\star(N) \to \lambda$ as $N \to \infty$ when $\lambda > 0$ and $\lambda_\star(N) \to 0$ as $N \to D^-$ when $\lambda = 0$.*

Proof. Write $F(\mu, N) := \mu - \lambda - \frac{\mu}{N}\mathrm{df}(\mu)$, and recall that df is continuous and strictly decreasing from $\mathrm{df}(0^+) = D$ to $\mathrm{df}(\infty) = 0$. For $\lambda = 0$ a root $\mu > 0$ solves $\mathrm{df}(\mu) = N$, so it exists exactly when $N < D$, and $\lambda_\star = \mathrm{df}^{-1}(N)$ is unique, strictly decreasing in N, and tends to 0 as $N \to D^-$.

For $\lambda > 0$, $F(\mu, N) < 0$ for $0 < \mu \leq \lambda$, and $F(\mu, N) \to \infty$ as $\mu \to \infty$ because $\mu\,\mathrm{df}(\mu) < \sum_k \zeta_k$, so a root $\lambda_\star > \lambda$ exists. At any root $\mathrm{df}(\lambda_\star) = N(1-\lambda/\lambda_\star) < N$, and since $\zeta^2/(\zeta+\mu)^2 < \zeta/(\zeta+\mu)$,

$$\left.\frac{\partial F}{\partial \mu}\right|_{\lambda_\star} = 1 - \frac{1}{N}\sum_k \frac{\zeta_k^2}{(\zeta_k + \lambda_\star)^2} \;>\; 1 - \frac{\mathrm{df}(\lambda_\star)}{N} \;>\; 0.$$

So F crosses zero upwards at every root, and the root is unique. Also $\partial F/\partial N = \frac{\lambda_\star}{N^2}\mathrm{df}(\lambda_\star) > 0$, so by the implicit function theorem $d\lambda_\star/dN = -(\partial F/\partial N)/(\partial F/\partial \mu) < 0$. Finally, $\lambda_\star \leq \lambda_\star(N_0)$ for $N \geq N_0$, and $0 < \lambda_\star - \lambda = \frac{\lambda_\star}{N}\mathrm{df}(\lambda_\star) \leq \lambda_\star(N_0)D/N \to 0$. □

Proposition 6 (Claim 2 under squared loss) *Under Assumptions 3 and 4, Assumption 1(a) and availabilities separated across the blocks, and with $Cn < D$ when $\lambda = 0$, the recovered fractions at n images per class are*

$$L_q(n) = \frac{1}{K}\sum_{j \in q} \frac{\zeta_j}{\zeta_j + \lambda_\star(Cn)} + o(1), \tag{55}$$

every L_q is strictly increasing in n, rising towards $\frac{1}{K}\sum_{j\in q}\zeta_j/(\zeta_j+\lambda)$ as $n \to \infty$ when $\lambda > 0$ and towards 1 as $Cn \to D$ when $\lambda = 0$, and

$$t_\star(n)^2 = \frac{L_{hi}(n)}{L_{lo}(n)} \quad \textit{is strictly decreasing in } n, \qquad 1 < t_\star(n)^2 \leq \frac{\zeta_{hi}}{\zeta_{lo}}. \tag{56}$$

Proof. The last two steps of the proof of Lemma 4 use the fit only through the recovery factor of each direction, which Assumption 4 supplies; this gives Eq. 55, with $\lambda_\star(Cn)$ well defined by Lemma 6. Each summand is strictly decreasing in $\lambda_\star$, and $\lambda_\star$ is strictly decreasing in $N = Cn$ with the limits of Lemma 6, so each L_q is strictly increasing in n with the stated limits. The empirical fit is linear in φ, so Lemma 3 gives $t_\star^2 = L_{hi}/L_{lo}$, which is strictly increasing in $\lambda_\star$ by Corollary 1 and hence strictly decreasing in n. The bracket is Lemma 5 at $\lambda_\star > 0$. □

Proposition 6 is Claim 2, and it also supplies the mechanism the claim asserts. What rises with the budget is the recovered fraction $\zeta_j/(\zeta_j+\lambda_\star(Cn))$ in every direction, and it saturates at 1; a direction of low availability starts further from saturation and therefore gains proportionally more, which is why the ratio falls. The linearised ConvNet trained with cross-entropy behaves this way: between IPC-5 and IPC-500 in Table 26 (without CCJ), L_{lo} rises by a factor of 5.4 and L_{hi} by a factor of 1.6.

This is why Claim 2 is stated in terms of the recovered fraction rather than the availability: ζ_j is a property of the kernel and the background distribution and does not depend on n at any point above. What the budget raises is the *recovered* fraction of each feature, through $\lambda_\star$; availability is the fixed quantity that determines how fast that fraction rises.

F.6 Both claims need a regulariser

The two propositions above exclude $\lambda = 0$ with $N \geq D$, and they must.

Proposition 7 (ridgeless interpolation is blind to availability) *Let $\lambda = 0$ and $N \geq D$ with Ψ of full column rank. Then the fitted function of Eq. 43 does not depend on $\{\zeta_j\}$, every recovered fraction equals one in expectation, and $t_\star = 1$: both claims are vacuous.*

Proof. The ridgeless solution is $\hat{w} = (\Psi^\top \Psi)^{-1} \Psi^\top Y$ and the fitted function is $\hat{f}(x) = \psi(x)^\top \hat{w}$. Rescaling the features by any invertible diagonal R sends $\Psi \mapsto \Psi R$ and $\hat{w} \mapsto R^{-1}\hat{w}$, leaving $\hat{f}$ unchanged. Since $\psi_j = \sqrt{\zeta_j}\varphi_j$, the fit is therefore the ordinary least squares fit in the φ coordinates, whatever the availabilities are. By Lemma 1, expanding the softmax for weak features, $\mathbb{E}[Y_c \mid X]$ is linear in φ at leading order in $\|d\|$, with the $\lambda = 0$ coefficients of Eq. 49. Least squares is unbiased for a linear conditional mean, so every recovery factor is one, $L_{hi} = L_{lo} = 1$, and Lemma 3 gives $t_\star = 1$. □

So availability shapes which feature the learner prefers only because training is stopped early, explicitly regularised, or run in the overparameterised regime where the design itself regularises. The three are the same mechanism in this model: $\tau < \infty$, $\lambda > 0$, and $\lambda = 0$ with $Cn < D$ all give a strictly positive effective regularisation, and Corollary 1 treats them alike.

The third case is the one that matters here, because it needs no regulariser to be chosen. When $Cn < D$ the ridgeless estimator carries $\lambda_\star > 0$ solving $\mathrm{df}(\lambda_\star) = Cn$, and Lemma 6 makes it fall with the budget: the coreset regime supplies its own regularisation, and both claims hold there with λ set to zero. This yields a prediction the propositions did not need but which is testable: $t_\star \to 1$ as Cn approaches the number of kernel eigendirections carrying appreciable availability, so the preference for high-variance features should disappear once the budget reaches the effective dimension of the kernel, and not before.

The measurements in Table 26 and Figure 2 are run under unregularised cross-entropy with a fixed step budget, so the operative regulariser there is early stopping rather than either ridge. This appendix is a parallel derivation of the two claims in a setting where they can be proved, not a model of those particular runs.

F.7 What the claims need

Collecting the hypotheses, Claim 1 holds whenever

- (C1) the blocks are separated in the covariance spectrum, $\lambda_j > \lambda_k$ for $j \in hi$ and $k \in lo$, which the construction Eq. 1 arranges;
- (C2) the kernel gains do not reverse that ordering, $\lambda_j g_j > \lambda_k g_k$, where $\zeta_j = \lambda_j g_j$ by Proposition 3;
- (C3) Assumption 1(a) and (c) hold, so that Stein's identity applies;
- (C4) the learner is regularised, early stopped, or overparameterised, in the sense of Section F.6.

Claim 2 needs (C1)–(C4) and in addition

- (C5) the regularisation is held fixed as the budget varies. If the stopping time is set in epochs rather than in steps, or the ridge is tuned per budget, $\lambda_\star$ need not be monotone in n and the claim can fail;
- (C6) the budget stays below the effective dimension, $Cn < \mathrm{df}(0^+)$, in the ridgeless case; beyond it Proposition 7 applies and $t_\star = 1$.

(C2) is the only one of these that the experiments cannot check directly, and it is the assumption on which Claim 1 genuinely rests: nothing forbids an architecture whose sensitivities g_j fall faster than the variances λ_j rise, and in such a model the low-variance block would be the more available one and the preference would reverse.

G Evaluation Protocol

Table 30 summarizes the setting which was used to evaluate the main tables Table 1 and Table 2. Every number reported is top-1 accuracy on the unmodified test split of the dataset. No jitter or

any other augmentation is applied to the test inputs. We trained over three seeds. The full-dataset row of Table 2 is marked † because it is the published top-1 accuracy of torchvision's pretrained ResNet-18 (TorchVision maintainers and contributors, 2016), trained under torchvision's own reference recipe rather than the protocol of Table 30, and we quote it only to place the coreset numbers on the scale of full-data training.

Table 30: **The two training protocols.**

	CIFAR-100 (main)	ImageNet-1K
student	ResNet-18	ResNet-18 (torchvision)
resolution	32 px	224 px
epochs	200	300
batch	32	16
learning rate	0.1	0.01
momentum	0.9 (Nesterov)	0.9
weight decay	5×10^{-4}	10^{-4}
schedule	cosine, per step	$\times 0.2$ at $\frac{2}{3}$, $\frac{5}{6}$
augmentation	crop, flip	RRC, flip, CutMix
coreset	16 selectors	11 selectors
seeds	42, 43, 44	0, 1, 2

Table 8 reads no label before the annotation step: DINOv2 (Oquab et al., 2024) ViT-S/14 features of the unlabelled training set, spherical k-means into $k = C$ pseudo-classes, ELFS-style selection inside each pseudo-class, and a ConvNet-3 trained for 500 epochs under crop, flip and colour jitter. The CCJ pooled variances are rebuilt over pseudo-clusters rather than classes.

Scores are computed with the released code of each selector, and the resulting mask is *frozen* before any student is trained. Forgetting, GraNd, least-confidence, margin, CAL and DeepFool are selected after trimming the easiest 20% of the pool under AUM; the other selectors apply each score's own top-k rule to the untrimmed pool. If a selector has a hyperparameter, it is selected on a held-out split of the *training* set. CCS, D^2 and herding use the hyperparameters as described in their respective papers.

H Compute Cost

We measure what CCJ and QDA cost on the code and recipes of Table 1 and Table 2, on one RTX 3090 per dataset (Table 31). As in those tables, `std` trains the same student on the same coreset without CCJ. QDA does not change training, so its cost appears only in the lower rows: estimating the class statistics of Eq. 13 once per dataset, fitting T_{cal} once per coreset, and evaluating the QDA at test time, reported as a percentage of the network's own inference time.

Table 31: **Compute cost of CCJ and QDA.**

	CIFAR-100		ImageNet-1K	
	`std`	+CCJ	`std`	+CCJ
training				
seconds per epoch, IPC-10	0.447	0.465	11.05	11.34
images per second, IPC-10	2,237	2,150	905	882
seconds per epoch, IPC-50	1.627	1.682	55.07	56.26
images per second, IPC-50	3,073	2,972	908	889
peak GPU memory (MiB)	1,429	1,429	372	372
once per dataset				
σ_a^2	0.16 s		22.7 min	
$A\mu_c$ and S_c^{dct}	0.13 s		22.5 min	
once per coreset				
T_{cal}	0.25–0.27 s		1.6–1.8 s	
QDA at test time (% of network)	0.8%		3.0%	