Escaping Parameter Space: Tight Generalization Bounds via Representation Quality

Niclas Göring[1]* **Shuofeng Zhang**[2] **Branton DeMoss**[3] **Ard A. Louis**[1]

[1]Rudolf Peierls Centre for Theoretical Physics, University of Oxford, Oxford, UK
[2]Meta
[3]Mathematical Institute, University of Oxford, Oxford, UK

Abstract

We derive a tight generalization bound for deterministic neural We derive a tight generalization bound for deterministic neural networks based on their last hidden layer representations. By evaluating representation quality with a parameter-free k-nearest-neighbor (k-NN) classifier, our certificate avoids traditional parameter-space complexity measures, KL divergences to a posterior, weight norms, and compression. The bound is tight: trained from scratch, it reaches an 8.0 percentage point gap to test error on CIFAR-10 (ResNet-18) and yields the first non-vacuous certificates we are aware of on CIFAR-100; with frozen DINOv2 representations, it certifies at 1.4% on CIFAR-10 and 23.1% on 1,000-class ImageNet. It decomposes additively into three empirical, independently measurable quantities: local class geometry (k-NN decoder error), decoder-network agreement, and training stability. This allows the bound to act as a diagnostic tool: we demonstrate that actively destroying local class geometry eliminates generalization while leaving dataset memorization intact, identifying class geometry as a causally relevant mechanism for generalization. Finally, we show that learned representations are significantly less sensitive to weight noise than linear readouts, suggesting why parameter-space certificates often struggle to remain tight.

1 Introduction

Almost every existing non-vacuous generalization bound for deep networks pays a price for their treatment of parameter space. PAC-Bayes certificates add noise to the weights and pay through KL divergence Dziugaite and Roy [2017], Pérez-Ortiz et al. [2021], Lotfi et al. [2022]. Compression bounds quantize or prune and pay through description length Zhou et al. [2019], Lotfi et al. [2022]. Recent work has produced non-vacuous certificates for deterministic networks by reasoning about partitions Than and Phan [2025] or distribution and loss complexity. Each line of work has produced progressively tighter results, but all share a common design: they reason about parameters, or parameter-derived quantities, to certify predictions.

We take a different route. Instead of certifying the parameterized network through its weights, we certify it through its learned representation. Concretely, we produce two bounds: in the transfer setting, where the feature map is fixed independently of the target data, we certify a k-nearest-neighbor (k-NN) classifier built on the last-hidden-layer representations $h^{(L)}(\boldsymbol{x})$; in the cross-fit setting, where the network is trained on the target task, we certify the deployed deterministic network itself, using the k-NN decoder. The proof uses only cross-fitting and the KL concentration inequality for i.i.d. Bernoulli variables. The representation is of course produced by the parameters, but no parameter-space complexity measure, no KL to a posterior, weight norm, covering number, or description length

*Corresponding author: niclas.goring@physics.ox.ac.uk

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

enters the argument. The bound is a function of three empirical quantities on the learned feature map $\varphi\colon \boldsymbol{X} \to \mathcal{Z}$ (Figure 1a):

- **Q1** *Can a simple classifier in $\mathcal{Z}$ predict labels?* That is, does the representation have clean local class geometry?
- **Q2** *Does that classifier agree with the network?* That is, does the representation capture what the network actually computes?
- **Q3** *Is training stable under fold removal?* That is, do models trained on n versus $n - n_j$ points make essentially the same predictions?

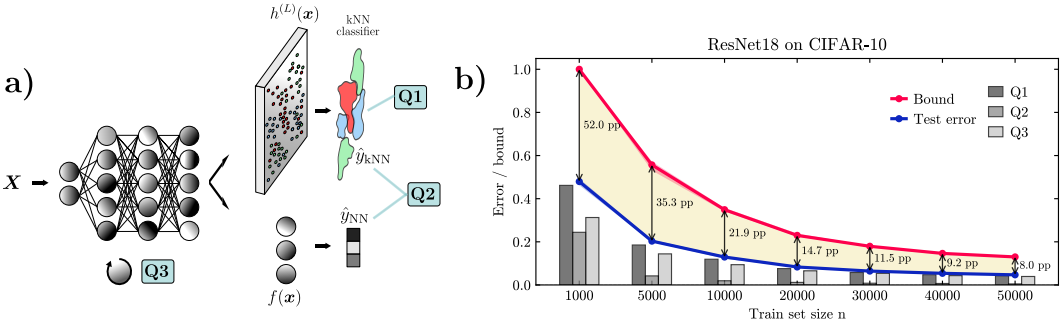


Figure 1: Any learned feature map $\varphi\colon \boldsymbol{X} \to \mathcal{Z}$ raises three questions: **Q1** Can a simple classifier in $\mathcal{Z}$ predict labels? **Q2** Does it agree with the network? **Q3** Is training stable under fold removal? **a)** A k-NN classifier on the last-hidden-layer representation $h^{(L)}(\boldsymbol{x})$ yields $\hat{y}_{\mathrm{kNN}}$ (**Q1**), compared against $\hat{y}_{\mathrm{NN}}$ (**Q2**); a fold-trained model tests stability (**Q3**). **b)** ResNet-18 on CIFAR-10: the bound converges to within 8.0 pp of test error at n=50,000, the tightest known certificate from a deterministic network without modified training. Stacked bars show **Q1** dominates: representation quality is the binding constraint.

When all three answers are yes, the deterministic deployed network is certified without ever reasoning about its parameters. The bound decomposes additively into one term per question, turning the certificate into a diagnostic of representation quality. Our **contributions** are:

1. **A representation-level certificate for deterministic networks** (Theorem 2.6). Three empirical terms, decoder error (**Q1**), decoder–network disagreement (**Q2**), and training stability (**Q3**) bound the test error of the deployed network, without any parameter-space KL term, weight norm, covering number, or compression in the proof.
2. **Tight bounds across scales.** Trained from scratch: gaps to test error as small as 8.0 pp obtain on CIFAR-10, and the first non-vacuous certificates we are aware of on CIFAR-100 for unmodified deterministic networks. Frozen pretrained representations achieve 1.4% on CIFAR-10 and 23.1% on 1,000-class ImageNet (DINOv2), the first non-vacuous k-NN decoder certificates we are aware of at ImageNet scale. See Tables 1 and 2.
3. **Implications of the decomposition.** The three terms can be measured independently, which localizes where generalization succeeds or fails. Double descent lives in the linear readout (**Q2**), not the representation (**Q1**); see Section 4.1. An intervention that destroys local class geometry eliminates generalization while training accuracy stays at 100%, interventional evidence that **Q1** is causal, not a correlate of generalization (Section 4.2).

1.1 Related work

Most non-vacuous certificates for deep networks reason through parameter-space quantities, whether via PAC-Bayes perturbation of the weights [McAllester, 1998, Dziugaite and Roy, 2017, Neyshabur et al., 2017, Pérez-Ortiz et al., 2021] or compression and quantization [Hinton and Van Camp, 1993, Arora et al., 2018, Zhou et al., 2019, Lotfi et al., 2022]. Our bound takes a different route: no parameter-space complexity measure enters the certificate, and its tightness depends on representation quality rather than parameter count (Section 4.1). Function-space PAC-Bayes analyses also move beyond parameter space [Valle-Pérez and Louis, 2020, Zhang and Louis, 2025]; our approach is not the first to do so, but requires only held-out error, agreement, and stability. Two recent methods are

related: [Than and Phan, 2025] certify deterministic networks via an a priori data partition; [Bazinet et al., 2026] certify via a separately trained surrogate and empirical disagreement. Our surrogate is neither trained nor chosen: it is a parameter-free k-NN decoder induced by the representation itself, and the bridge to the deployed network reduces to training stability [Bousquet and Elisseeff, 2002, Hardt et al., 2016]. Our proof uses cross-fit held-out evaluation with KL concentration [Langford and Schapire, 2005, Blum et al., 1999]; our contribution is not the technique but the object being evaluated. k-NN probing of representations is standard in self-supervised learning [Wu et al., 2018, Caron et al., 2021, Oquab et al., 2023]; our contribution is that probe performance, combined with agreement and stability, becomes a formal certificate for the deployed network. We compare numerical values of our certificates in Tables 1 and 2 and discuss related work in more depth in Appendix B.

2 Theory: A Representation-Level Bound

Any learned feature map $\boldsymbol{\varphi}\colon \boldsymbol{X} \to \mathcal{Z}$ turns a dataset into a point cloud in some metric space $(\mathcal{Z}, d)$. In practice, $\boldsymbol{\varphi}$ is the network's last hidden layer representation $h^{(L)}(\boldsymbol{x})$, the vector fed into the final linear classifier, and d is ℓ_2 distance; extraction requires a single forward pass with no modification to the network (details in Appendix F). The theorems apply verbatim to any deterministic feature map (e.g. another layer of the network) and metric d. We instantiate the three questions from the introduction by taking the "simple classifier" to be a k-NN rule in $(\mathcal{Z}, d)$. It requires no training, has no parameters of its own, and makes all three quantities directly measurable. We emphasize what this does and does not mean: the representation $\boldsymbol{\varphi}$ is of course produced by the trained parameters, but the bound uses no parameter-space complexity measure. The proofs use a single structural property: conditioned on the complement sample, the held-out points are i.i.d. and the decoder is deterministic, so standard concentration applies.

2.1 Technical setup

Let P be a distribution on $\boldsymbol{X} \times [C]$, where $[C] \coloneqq \{1, \dots, C\}$. Draw an i.i.d. sample $\mathcal{S}_n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \sim P^n$. Let $\mathcal{A}$ be a (possibly randomized) training algorithm that maps any labeled dataset $\mathcal{T}$ to network parameters $\mathcal{A}(\mathcal{T})$, producing a classifier $\widehat{f}_{\mathcal{A}(\mathcal{T})}$ and a feature map $\boldsymbol{\varphi}_{\mathcal{A}(\mathcal{T})}\colon \boldsymbol{X} \to \mathcal{Z}$ into a metric space $(\mathcal{Z}, d)$. All internal randomness of $\mathcal{A}$ (initialization, data ordering, augmentation) is fixed in advance; every statement below is conditional on this randomness. Write $\boldsymbol{\theta}^{(\mathrm{full})} = \mathcal{A}(\mathcal{S}_n)$ for the full-data parameters, $\widehat{f}^{(\mathrm{full})}$ for the deployed classifier, and $\boldsymbol{\varphi}^{(\mathrm{full})} = \boldsymbol{\varphi}_{\boldsymbol{\theta}^{(\mathrm{full})}}$ for its feature map. This is the object we wish to certify.

Definition 2.1 (Cross-fit k-NN decoder)**.** Partition $[n]$ into J disjoint folds $F_1, \dots, F_J$ with $|F_j| = n_j$ and $\sum_j n_j = n$. For each fold j, train on the complement:

$$\boldsymbol{\theta}^{(-j)} = \mathcal{A}\big(\mathcal{S}_{-j}\big), \qquad \mathcal{S}_{-j} = \big\{(\boldsymbol{x}_i, y_i) : i \notin F_j\big\}.$$

Write $\boldsymbol{\varphi}^{(-j)}$ for the fold-j feature map, $\boldsymbol{z}_i^{(-j)} = \boldsymbol{\varphi}^{(-j)}(\boldsymbol{x}_i)$ for the embedded points, and $\widehat{f}_j = \widehat{f}_{\boldsymbol{\theta}^{(-j)}}$ for the fold-j classifier. The *cross-fit k-NN decoder* $h_j^{(k)}$ maps a query $\boldsymbol{x}$ to the majority label among the k nearest neighbors of $\boldsymbol{\varphi}^{(-j)}(\boldsymbol{x})$ in the reference set $\{\boldsymbol{z}_i^{(-j)} : i \notin F_j\}$, with distance d. All tie-breaking rules are fixed in advance and deterministic.

Three empirical quantities control the certificate:

	Definition	Measures
$\widehat{R}_j^{(k)}$	$\frac{1}{n_j}\sum_{i\in F_j} \mathbf{1}[h_j^{(k)}(\boldsymbol{x}_i) \neq y_i]$	**Q1**: decoder error
$\widehat{D}_j^{(k)}$	$\frac{1}{n_j}\sum_{i\in F_j} \mathbf{1}[h_j^{(k)}(\boldsymbol{x}_i) \neq \widehat{f}_j(\boldsymbol{x}_i)]$	**Q2**: decoder–network gap
Δ_j^{stab}	$\frac{1}{n_j}\sum_{i\in F_j} \mathbf{1}[\widehat{f}_j(\boldsymbol{x}_i) \neq \widehat{f}^{(\mathrm{full})}(\boldsymbol{x}_i)]$	**Q3**: fold–full gap

Note that Δ_j^{stab} is independent of k: it compares the two networks' hard predictions directly.

Notation. The binary relative entropy is $\mathrm{kl}(q\|p) = q\log\frac{q}{p} + (1-q)\log\frac{1-q}{1-p}$ for $q, p \in [0,1]$ (extended by continuity at the boundary), and its upper inverse is $\mathrm{kl}_+^{-1}(q,c) = \sup\{p \in [q,1] : \mathrm{kl}(q\|p) \le c\}$.

2.2 Certifying the decoder (Q1)

The first step certifies the k-NN decoder alone, treating it as a classifier in its own right. Because the fold points F_j are held out from training $\boldsymbol{\theta}^{(-j)}$, the decoder is conditionally deterministic and the error indicators are conditionally i.i.d. Hence, a direct application of KL concentration suffices.

Theorem 2.2 (Cross-fit decoder certificate)**.** *Fix a finite search set $\mathcal{K} \subset \mathbb{N}$ (e.g. $\mathcal{K} = \{1,3,5\}$). With probability at least $1-\delta$ over $\mathcal{S}_n$, for every $j \in [J]$ and $k \in \mathcal{K}$ simultaneously:*

$$\Pr_{(\boldsymbol{X},Y)\sim P}\big(h_j^{(k)}(\boldsymbol{X}) \neq Y \mid \mathcal{S}_{-j}\big) \;\le\; \mathrm{kl}_+^{-1}\left(\widehat{R}_j^{(k)},\;\; \frac{\log\big(|\mathcal{K}|\,J/\delta\big) + \log(n_j{+}1)}{n_j}\right). \tag{1}$$

Since $h_j^{(k)}$ depends on $\mathcal{S}_n$ only through $\mathcal{S}_{-j}$, conditioning on $\mathcal{S}_{-j}$ is equivalent to conditioning on $\mathcal{S}_n$.

Proof. See Section D.1 for the proof. □

2.3 Transfer certificate

When the feature map $\boldsymbol{\varphi}$ is learned without access to the target data, e.g. pretrained on a different dataset, the cross-fit construction simplifies. There is no fold/full distinction, so **Q3** vanishes, **Q2** is irrelevant (the k-NN decoder *is* the deployed classifier), and a single reference/evaluation split replaces the J-fold construction. For example, this lets us certify a k-NN decoder built from an ImageNet-pretrained ResNet-50 representation on CIFAR-10; see Section 3.1.

Theorem 2.3 (Transfer certificate)**.** *Let $\boldsymbol{\varphi}$ be fixed independently of the target sample $\mathcal{S}_n$ (e.g. pretrained on disjoint data). Split $\mathcal{S}_n \sim P^n$ into $\mathcal{S}_{\mathrm{ref}}$ (size m) and $\mathcal{S}_{\mathrm{eval}}$ (size $n_e = n-m$). Let $I_{\mathrm{eval}} \subset [n]$ denote the indices of the evaluation points. Build $h^{(k)}$ from $\boldsymbol{\varphi}(\mathcal{S}_{\mathrm{ref}})$. Define $\widehat{R}^{(k)}$ as in Section 2.1 but computed on $\mathcal{S}_{\mathrm{eval}}$. With probability $\ge 1-\delta$ over $\mathcal{S}_n$, for every $k \in \mathcal{K}$ simultaneously:*

$$\Pr_{(\boldsymbol{X},Y)\sim P}\big(h^{(k)}(\boldsymbol{X}) \neq Y \mid \mathcal{S}_{\mathrm{ref}}\big) \;\le\; \mathrm{kl}_+^{-1}\big(\widehat{R}^{(k)},\; c_e\big), \tag{2}$$

where $c_e = \big[\log(|\mathcal{K}|/\delta) + \log(n_e{+}1)\big]/n_e$.

Q2 $= 0$ and **Q3** $= 0$ by construction: the k-NN decoder is the classifier and $\boldsymbol{\varphi}$ does not depend on the target data. Eq. (2) certifies the decoder via **Q1** alone.

Proof. See Section D.2 for the proof. □

2.4 Certifying the full network

When the network is trained on the target task, **Q2** and **Q3** are both non-trivial. We handle them in two steps. We bound the deployed network $\widehat{f}^{(\mathrm{full})}$ without accessing its parameters: its error is at most (a) the error of a surrogate fold network plus (b) the disagreement between fold and full networks on unlabeled data.

Step A: from decoder to fold network.

Theorem 2.4 (Fold-network certificate)**.** *Fix $\mathcal{K} \subset \mathbb{N}$. With probability at least $1-\delta$ over $\mathcal{S}_n$, for every $j \in [J]$ and $k \in \mathcal{K}$ simultaneously:*

$$\Pr_{(\boldsymbol{X},Y)\sim P}\big(\widehat{f}_j(\boldsymbol{X}) \neq Y \mid \mathcal{S}_{-j}\big) \;\le\; \underbrace{\mathrm{kl}_+^{-1}\big(\widehat{R}_j^{(k)},\; c_j\big)}_{\textbf{Q1}} \;+\; \underbrace{\mathrm{kl}_+^{-1}\big(\widehat{D}_j^{(k)},\; c_j\big)}_{\textbf{Q2}} \tag{3}$$

where $c_j = \big[\log(2\,|\mathcal{K}|\,J/\delta) + \log(n_j{+}1)\big]/n_j$. *Since* $\widehat{f}_j$ *depends on* $\mathcal{S}_n$ *only through* $\mathcal{S}_{-j}$, *conditioning on* $\mathcal{S}_{-j}$ *is equivalent to conditioning on* $\mathcal{S}_n$.

Proof. See Section D.3 for the proof. □

Step B: from fold network to full-data network. For any $(\boldsymbol{x}, y)$ and $j \in [J]$,

$$\mathbf{1}\big[\widehat{f}^{(\mathrm{full})}(\boldsymbol{x}) \neq y\big] \;\leq\; \mathbf{1}\big[\widehat{f}_j(\boldsymbol{x}) \neq y\big] \;+\; \mathbf{1}\big[\widehat{f}_j(\boldsymbol{x}) \neq \widehat{f}^{(\mathrm{full})}(\boldsymbol{x})\big], \tag{4}$$

since the full network can only err if the fold network also errs or the two disagree. Taking expectations conditional on $\mathcal{S}_n$ introduces the *stability gap*

$$\Delta_j^{\mathrm{stab}} \;=\; \Pr_{\boldsymbol{X}\sim P_X}\big(\widehat{f}_j(\boldsymbol{X}) \neq \widehat{f}^{(\mathrm{full})}(\boldsymbol{X}) \mid \mathcal{S}_n\big). \tag{5}$$

This is a disagreement probability over $\boldsymbol{X}$ alone; labels play no role, so it can be certified from unlabeled data. Rather than derive an optimizer-specific analytic stability bound that would need to be rewritten for SGD, Adam, RAdam, Lion, Shampoo, or any new optimizer, we certify this term empirically. This keeps the certificate practically usable across training setups.

Theorem 2.5 (Direct stability certification)**.** *Let* $\mathcal{U} = \{\boldsymbol{x}_r\}_{r=1}^m \sim P_X^m$ *be independent of* $\mathcal{S}_n$.[2] *Define* $\widehat{\Delta}_j^{\mathcal{U}} = \frac{1}{m}\sum_{r=1}^m \mathbf{1}[\widehat{f}_j(\boldsymbol{x}_r) \neq \widehat{f}^{(\mathrm{full})}(\boldsymbol{x}_r)]$. *With probability* $\geq 1-\delta'$ *over the joint draw* $(\mathcal{S}_n, \mathcal{U})$, *for every* $j \in [J]$ *simultaneously:*

$$\Delta_j^{\mathrm{stab}} \;\leq\; \mathrm{kl}_+^{-1}\left(\widehat{\Delta}_j^{\mathcal{U}},\; \frac{\log(J/\delta') + \log(m{+}1)}{m}\right) \;=:\; B_j^{\mathrm{stab}}. \tag{6}$$

Proof. See Section D.4 for the proof. □

Combining Theorem 2.4, (4), and Theorem 2.5:

Theorem 2.6. *Let* $\mathcal{U} = \{\boldsymbol{x}_r\}_{r=1}^m \sim P_X^m$ *be independent of* $\mathcal{S}_n$. *With probability* $\geq 1-\delta-\delta'$ *over* $(\mathcal{S}_n, \mathcal{U})$, *for every* $k \in \mathcal{K}$ *and* $j \in [J]$:

$$\Pr_{(\boldsymbol{X},Y)\sim P}\big(\widehat{f}^{(\mathrm{full})}(\boldsymbol{X}) \neq Y \mid \mathcal{S}_n\big) \;\leq\; \underbrace{\mathrm{kl}_+^{-1}\big(\widehat{R}_j^{(k)},\, c_j\big)}_{\substack{\textbf{Q1}\text{: local class}\\ \text{geometry}}} \;+\; \underbrace{\mathrm{kl}_+^{-1}\big(\widehat{D}_j^{(k)},\, c_j\big)}_{\substack{\textbf{Q2}\text{: decoder–}\\ \text{network agreement}}} \;+\; \underbrace{\mathrm{kl}_+^{-1}\big(\widehat{\Delta}_j^{\mathcal{U}},\, c_m\big)}_{\substack{\textbf{Q3}\text{: training}\\ \text{stability}}} \tag{7}$$

where $c_j = \big[\log(2\,|\mathcal{K}|\,J/\delta) + \log(n_j{+}1)\big]/n_j$ *and* $c_m = \big[\log(J/\delta') + \log(m{+}1)\big]/m$. *The tightest certificate is* $\min_{j,k}\big[\mathrm{kl}_+^{-1}(\widehat{R}_j^{(k)}, c_j) + \mathrm{kl}_+^{-1}(\widehat{D}_j^{(k)}, c_j) + \mathrm{kl}_+^{-1}(\widehat{\Delta}_j^{\mathcal{U}}, c_m)\big]$.

Interpretation

- **Q1** $\widehat{R}_j^{(k)}$ small $\iff$ the representation has clean local class geometry.
- **Q2** $\widehat{D}_j^{(k)}$ small $\iff$ the network computes through this representation.
- **Q3** Δ_j^{stab} small $\iff$ the training algorithm is stable under fold removal.

Proof. See Section D.5 for the proof. □

The representation φ and metric d enter the bound only through the three empirical quantities $\widehat{R}_j^{(k)}$, $\widehat{D}_j^{(k)}$, and $\widehat{\Delta}_j^{\mathcal{U}}$; no parameter-space complexity measure appears anywhere. What makes the bound tight is not the proof technique but the learned representation, which creates a local class geometry so clean that a trivial classifier nearly matches the network. The bound measures that quality; it does not create it.

[2] The standard test inputs of MNIST, CIFAR-10, and CIFAR-100 satisfy this: no test point participates in training any fold or full model.

3 Results

We evaluate the certificate in two settings: transfer with zero target training (Section 3.1) and cross-fit certification of networks trained from scratch (Section 3.2), then compare with prior bounds. All bounds use ℓ_2 distance on the last-hidden-layer representation $h_{\boldsymbol{\theta}} : \boldsymbol{X} \to \mathbb{R}^D$, the vector fed into the final linear classifier, computed by a single unmodified forward pass. For ResNets, $h_{\boldsymbol{\theta}}$ is the output of global average pooling (D=512 or 2048); for Vision Transformers, the class token after the final layer norm (D=768 or 1024); for DINOv2, the class-token embedding from the frozen backbone. Full extraction details are in Section F.

3.1 Transfer certification without training

When the feature map is fixed independently of the target data (e.g. pretrained on ImageNet), **Q2** and **Q3** vanish and Theorem 2.3 applies directly. The certified object is the k-NN decoder built on top of the frozen representation, not the pretrained network's own classifier head. We evaluate eight pretrained models plus a randomly initialized ResNet-18 as a null control. For each, we extract the ℓ_2 representation with a single forward pass, split the target data 80/20 ($n_e \approx 10$k), and report the tightest bound over $k \in \{1, 3, 5, 7, 11, 15, 21\}$ at δ=0.05 (the reported certificates are best-of-grid; the union bound over k is already included). Four target datasets are used: CIFAR-10, CIFAR-100, ImageNet validation, and ImageNet-V2 (distribution shift). No model undergoes any target-task network training. Table 1 reports the results.

		CIFAR-10			CIFAR-100			ImageNet			ImageNet-V2		
Model	Params	Err	Bnd	Gap	Err	Bnd	Gap	Err	Bnd	Gap	Err	Bnd	Gap
DINOv2-L/14	304M	0.9	**1.4**	**0.5**	10.4	**12.1**	**1.7**	20.9	23.1	**2.2**	37.5	43.0	5.5
DINOv2-g/14	1.1B	—	—	—	—	—	—	21.2	23.5	**2.3**	37.7	43.2	5.5
DINOv2-B/14	86M	2.3	3.2	**0.9**	14.9	16.9	**2.0**	23.4	25.7	**2.3**	41.3	46.8	5.5
ViT-B/16	86.6M	6.8	8.3	**1.5**	28.1	30.5	**2.4**	21.6	23.8	**2.2**	39.2	44.7	5.5
ViT-L/16	304M	7.7	9.2	**1.5**	29.5	32.0	**2.5**	21.8	24.0	**2.2**	40.3	45.8	5.5
WRN-50-2	68.9M	10.1	11.8	**1.7**	34.3	36.9	**2.6**	24.1	26.4	**2.3**	47.6	53.1	5.5
ResNet-50	25.6M	12.1	13.9	**1.8**	36.7	39.3	**2.6**	26.5	28.9	**2.4**	49.4	55.0	5.6
ResNet-18	11.7M	18.0	20.1	**2.1**	43.6	46.2	**2.6**	44.1	46.8	**2.7**	70.2	75.1	**4.9**
ResNet-18 (rand.)	*11.7M*	*63.3*	*65.8*	**2.5**	*87.8*	*89.5*	**1.7**	*99.0*	*99.4*	**0.4**	*99.5*	*99.9*	**0.4**

Table 1: **Transfer decoder certification (ℓ_2, δ=0.05).** The certified object is the k-NN decoder, not the pretrained classifier head (**Q2**, **Q3** vanish by construction). Err: k-NN error (%). Bnd: certified bound (%). Gap: <5 5–15 15–30 30–50 >50 pp.

Three findings stand out. **(i)** First, the bounds are tight: DINOv2-L/14 certifies at 1.4% on CIFAR-10 and 12.1% on CIFAR-100; all eight pretrained models produce non-vacuous certificates on 1,000-class ImageNet, the tightest being 23.1%. To our knowledge, these are the first non-vacuous ImageNet-scale certificates for k-NN decoders built from unmodified deterministic pretrained representations. **(ii)** Second, the certificate extends to distribution shift: all eight models also certify on ImageNet-V2 [Recht et al., 2019], with the wider gap (∼5.5 pp vs. ∼2.2 pp in-distribution) reflecting the harder setting rather than a limitation of the bound. **(iii)** Third, the bound tracks representation quality, not model size: DINOv2-g (1.1B params) and DINOv2-L (304M) produce nearly identical certificates on ImageNet (23.5% vs. 23.1%), while ResNet-18 pretrained vs. random, same 11.7M parameters, spans 20.1% to 65.8% on CIFAR-10. The certificate measures what the network learned, not how many parameters were used to express it.

3.2 Cross-fit certification of trained networks

We now certify networks trained from scratch using the full certificate (Theorem 2.6) with J=5 folds, δ=δ'=0.025, and $\mathcal{U}$ = standard test inputs. Table 2 reports the results alongside recent certificates from prior work.

On MNIST all architectures certify tightly (best: ResNet-18 at 2.6%); on CIFAR-10, ResNet-50 certifies at 12.4%; and on CIFAR-100, ResNet-50 certifies at 44.6%. To our knowledge, these are the first non-vacuous CIFAR-100 certificates we are aware of, and among the tightest CIFAR-10 certificates, for networks trained from scratch without modifying the certified network. The three-term decomposition shows *why*. ResNet-50 on CIFAR-10 splits as **Q1** = 5.7%, **Q2** = 1.8%, **Q3** = 1.2%:

clean geometry, faithful decoder, stable training. ViT on CIFAR-100 is vacuous and every term is large (**Q1** $= 43.7\%$, **Q2** $= 20.2\%$, **Q3** $= 35.4\%$). A small ViT on $40{,}000$ images does not learn a stable, class-coherent representation. The same architecture *pretrained* certifies at 30.5% (Table 1), with **Q3**$=0$ by construction: same theorem, only the representation changed. The bound diagnoses representation quality, not architecture: FFNNs certify tightly on MNIST (4.8%) and correctly vacuously on CIFAR ($> 95\%$, where they genuinely fail at 44% test error).

Method	Architecture	n	Test	Bound	Gap	Certified object
MNIST						
Dziugaite and Roy [2017]	FCN-600	60k	$\sim$2.5%	$\sim$17%	14.5	Stochastic net
Zhou et al. [2019]	LeNet-5	60k	$\sim$1%	$\sim$46%	45	Compressed net
Lotfi et al. [2022]	LeNet-5	60k	$\sim$1%	11.6%	10.6	Quant. subspace
Pérez-Ortiz et al. [2021]	FCN	60k	$\sim$1.4%	$\sim$2.3%	**0.9**	Stochastic net
Ours (FFNN)	FFNN	60k	1.5%	4.9%	**3.4**	**Det. net via repr.**
Ours (ViT)	ViT-small	60k	0.9%	3.5%	**2.6**	**Det. net via repr.**
Ours (ResNet)	ResNet-18	60k	**0.6%**	**2.6%**	**2.0**	**Det. net via repr.**
CIFAR-10						
Lotfi et al. [2022]	ConvNet	50k	–	$\sim$58.2%	–	Quant. subspace
Bazinet et al. [2026]	ResNet-18	50k	5.8%	17.36%	11.5	Det. net via surr.
Ours (FFNN)	FFNN	50k	44.6%	95.5%	50.9	**Det. net via repr.**
Ours (ViT)	ViT-small	50k	12.3%	31.9%	19.6	**Det. net via repr.**
Ours (ResNet-18)	ResNet-18	50k	4.7%	13.0%	8.3	**Det. net via repr.**
Ours (ResNet-50)	ResNet-50	50k	**4.3%**	**12.4%**	**8.0**	**Det. net via repr.**
CIFAR-100						
Lotfi et al. [2022]	ConvNet	50k	–	94.6%	–	Quant. subspace
Ours (FFNN)	FFNN	50k	75.2%	100.0%	24.8	**Det. net via repr.**
Ours (ViT)	ViT-small	50k	37.1%	99.3%	62.3	**Det. net via repr.**
Ours (ResNet-18)	ResNet-18	50k	21.2%	51.1%	29.9	**Det. net via repr.**
Ours (ResNet-50)	ResNet-50	50k	**19.4%**	**44.6%**	**25.2**	**Det. net via repr.**

Table 2: **Cross-fit certification vs. prior bounds.** Per-dataset blocks list prior work (above the rule) and our cross-fit bounds across architecture families (below, shaded). Best row per dataset in darker shade and bold. ReLU activation; SiLU results are in Table 3. ℓ_2 metric, $\delta{=}\delta'{=}0.025$, best over $k \in \{1, 3, 5, 7, 11, 15, 21\}$ and $j \in [5]$; union-bound cost included. Prior-work numbers from original publications.

Comparison with prior bounds. On MNIST our cross-fit bound (2.6%) is comparable to the tightest known certificate [Pérez-Ortiz et al., 2021] while certifying a deterministic network rather than a stochastic perturbation. On CIFAR-10, our ResNet-18 bound (13.0%) improves on [Bazinet et al., 2026] at the same architecture (17.36%); on CIFAR-100 we are not aware of prior non-vacuous certificates. The comparison is not entirely apples-to-apples: different methods certify different objects (stochastic nets, compressed nets, surrogates, deterministic nets, decoders), catalogued in Table 4. The distinctive property of our approach is that it certifies the deployed deterministic network without modified training and extends to pretrained representations.

4 What the bound reveals

4.1 Overparameterization dissolves at the representation level

Our bound does not index parameters, only the representation they produce. DINOv2-g (1.1B params) and DINOv2-L (304M) give nearly identical certificates on ImageNet (23.5% vs. 23.1%); ResNet-18 pretrained vs. random, same 11.7M parameters, spans 20.1% to 65.8% on CIFAR-10 (Table 1). The relevant variable is representation quality, not parameter count. The width-scaling experiment (Figure 2) makes this precise and yields a sharper claim: the bound decomposes double descent into a monotone representation term and a transient head–representation misalignment. We train ResNet-18 at width multipliers (w) $1\times$–$32\times$ on CIFAR-10 with 0%, 10%, and 20% label noise, and evaluate

each term at every width. To isolate the shape of each term we report its excess over the widest-network baseline, $\Delta Q_1(w) = Q_1(w) - Q_1(w_{\max})$ and analogously for ΔQ_2. ΔQ_1 is monotone in width: wider networks produce strictly cleaner local class geometry, with label noise shifting the curve upward without changing its shape. ΔQ_2 is the only non-monotone term, peaking sharply near the interpolation threshold and then collapsing. At the critical width the representation already supports accurate local classification, but the linear head fails to exploit this geometry. The classical double-descent curve in test error is therefore the sum of a monotone improvement in representation quality (Q_1) and a transient spike in head–representation misalignment (Q_2), both of which the certificate measures directly; the test-error hump is localized to Q_2. The overparameterization question carries an implicit assumption that parameter count is the relevant variable; our decomposition reframes it as: *why does training on natural data produce representations where simple local classifiers succeed?*

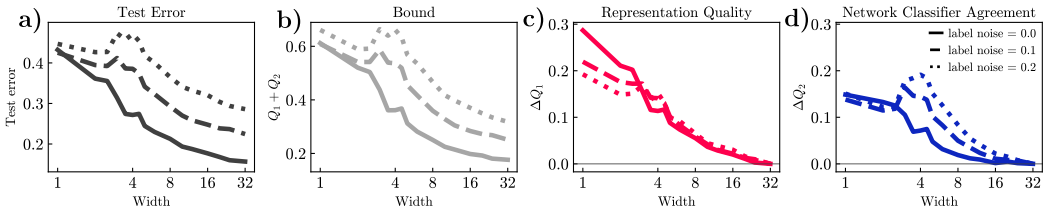


Figure 2: ResNet-18 width sweep on CIFAR-10 (0%, 10%, 20% label noise, 3 seeds). **a)** Test error and total bound. **b)** $\Delta Q_1(w)$ and $\Delta Q_2(w)$, each measured as excess over the widest-network baseline. **c)** ΔQ_1 (local class geometry) decreases monotonically in width. **d)** ΔQ_2 (decoder–network agreement) is the only non-monotone term, peaking at the interpolation threshold and then collapsing. The test-error hump is localized to Q_2. The Q_3 term is not analyzed in this sweep.

4.2 Local class geometry is causal, not correlational

The experiments above show that the bound *tracks* generalization. Does it measure a cause, or a symptom? A natural worry is that Q_1 and generalization are both downstream of some third factor (good optimization or a well-conditioned loss landscape, for example), in which case Q_1 would be a correlate rather than a mechanism. We test this with an intervention that alters Q_1 while leaving the network's predictions on the training set unconstrained, and ask whether generalization follows. Define a regularizer targeting exactly what Q_1 measures, within-class representation similarity:

$$\mathcal{L}_{\text{rep}} \;=\; \frac{1}{|\mathcal{P}|} \sum_{(i,j)\in\mathcal{P}} \Big(\bar{h}(\boldsymbol{x}_i)^\top \bar{h}(\boldsymbol{x}_j)\Big)^2, \qquad \mathcal{P} = \big\{(i,j) : i \neq j,\ y_i = y_j,\ i,j \in S\big\}, \tag{8}$$

where $\bar{h}(\boldsymbol{x}) = h(\boldsymbol{x})/(\|h(\boldsymbol{x})\|_2 + \varepsilon)$ is the ℓ_2-normalized representation at each hidden layer. Minimizing $\mathcal{L}_{\text{rep}}$ pushes same-class representations apart, destroying the local class geometry that Q_1 quantifies. Crucially, it places no constraint on the network's outputs: with enough capacity the network can reach $\mathcal{L}_{\text{rep}} \approx 0$ and $\mathcal{L}_{\text{CE}} = 0$ simultaneously, i.e. perfect training accuracy with destroyed class geometry, as shown ing Figure 3. We train six-layer FFNNs (width=1024) on SVHN with $\mathcal{L}_{\text{CE}} + \lambda\,\mathcal{L}_{\text{rep}}$ across four activation functions. Training accuracy saturates near 100% in every case: the network memorizes the data perfectly. Test accuracy strongly degrades. The trajectories differ across ReLU, SiLU, GELU, and tanh in both timing and the relative position of Q_1 and test error. But, the qualitative pattern is invariant: training stays at 100%, while test accuracy collapses as the Q_1 deteriorates. These results provide novel interventional evidence linking a certifiable quantity to generalization. By introducing a penalty that degrades the quality measure Q_1 without constraining the network's training predictions, we demonstrate that a generalizing network can be driven into a memorizing regime. Because this effect persists across the distinct optimization dynamics induced by four different activations, it strongly suggests that local class geometry acts as a fundamental mechanism separating generalization from mere memorization.

4.3 Why parameter-space certificates can be loose

PAC-Bayes bounds certify a stochastic classifier $\hat{f}_{\theta+\varepsilon}$ by adding noise to the weights. The bound is tight when the full network, including the linear readout, tolerates perturbation. Figure 4 suggests

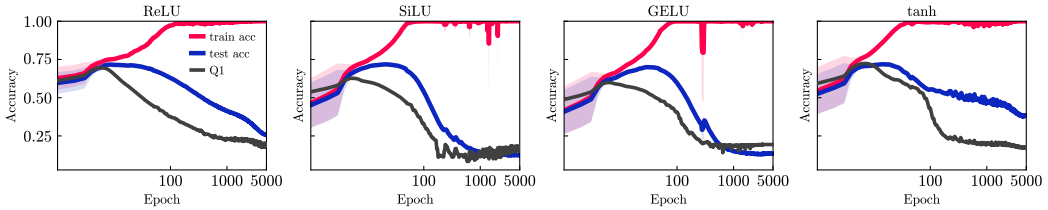


Figure 3: Six-layer FFNNs (width=1024) trained on SVHN with $\mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{rep}}$ across four activation functions. Training accuracy (blue) saturates near 100%; test accuracy (red) degrades toward chance ($\sim 10\%$). Q_1 (black) decays together with test error in every case. The qualitative Q_1-to-test-error coupling is invariant across activation functions, although trajectories differ quantitatively.

why this can be difficult: under Gaussian weight noise, k-NN accuracy in representation space degrades far more slowly than the network's own predictions. At σ=0.01 on a ReLU ResNet-18, the k-NN decoder retains 84% accuracy while the network has fallen to 67%. The pattern holds across activations and label-corruption levels (Figure 4b), with the gap ranging from $+2$ pp to $+46$ pp. The linear readout is consistently the more fragile component. This does not show that PAC-Bayes bounds cannot be tight; tighter variants exist [Pérez-Ortiz et al., 2021, Lotfi et al., 2022]. It offers an intuition for why certifying through parameters can pay for the fragility of the head, while certifying through the representation avoids this cost.

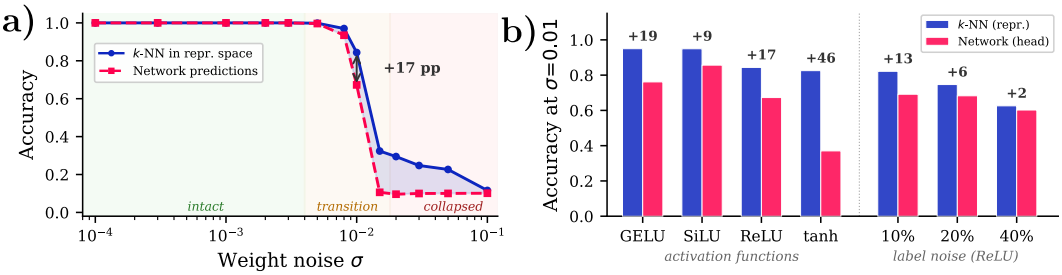


Figure 4: **a)** ReLU ResNet-18 on CIFAR-10. Under Gaussian weight noise at level σ, k-NN accuracy in ℓ_2 representation space (solid blue) degrades more slowly than the network's own predictions (dashed red). At σ=0.01 the k-NN decoder retains 84% accuracy while the network has fallen to 67%. **b)** The representation advantage holds across activation functions and label-corruption levels at σ=0.01, with the gap ranging from $+2$ pp (40% corruption) to $+46$ pp (tanh).

5 Discussion

The question we study is not: **Why do overparameterized networks generalize?**, but rather: **What determines good generalisation at the representation level?**. This is a question about representation learning and data structure, not capacity control. We find that good classification follows from clean local class geometry via elementary concentration. Our bound provides a quantitative tool for measuring this: a network with 10^8 parameters and one with 10^5 that induce the same feature geometry will receive the same bound.

Limitations. **Q3** is certified empirically rather than analytically. This ensures the bound remains usable across any optimizer (e.g., Adam), schedule, or augmentation without requiring re-derivation. The certificate specifically bounds classification error using last-layer ℓ_2 distance; exploring calibration, robustness, regression, or alternate metrics is left to future work. Finally, cross-fit certification requires training J fold models, a nontrivial one-off cost at ImageNet scale.

Broader impacts. While our certificates enable transparent evaluation by localizing generalization bottlenecks, the primary risk is over-trust. Our guarantee strictly bounds classification error under stated assumptions; it does not certify fairness, privacy, robustness, or overall deployment safety. Therefore, these bounds should inform, rather than unilaterally approve, high-stakes systems.

Future work. While our representation-level bound successfully decouples generalization from parameter-space capacity, it leaves open a crucial question: how does SGD reliably build this clean local class geometry in the first place? A natural next step is to investigate this mechanism through the emerging lens of "learning mechanics" [Simon et al., 2026], exploring whether the formation of certifiable class geometry can be derived as a predictable, macroscopic observable governed by the fundamental dynamics of the training process.
The code for the paper can be found in `https://github.com/NiclasGoering/escaping-parameter-space`.

Acknowledgments and Disclosure of Funding

This work was supported by the Laboratory for AI Security Research (LASR). The views expressed in this paper are those of the authors and do not necessarily reflect the position of LASR or His Majesty's Government.

References

Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *International conference on machine learning*, pages 254–263. PMLR, 2018.

Peter Bartlett, Yoav Freund, Wee Sun Lee, and Robert E Schapire. Boosting the margin: A new explanation for the effectiveness of voting methods. *The annals of statistics*, 26(5):1651–1686, 1998.

Peter L Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE transactions on Information Theory*, 44(2):525–536, 2002.

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of machine learning research*, 3(Nov):463–482, 2002.

Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. *Advances in neural information processing systems*, 30, 2017.

Mathieu Bazinet, Valentina Zantedeschi, and Pascal Germain. Bound to disagree: Generalization bounds via certifiable surrogates, 2026. URL `https://arxiv.org/abs/2602.23128`.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Avrim Blum, Adam Kalai, and John Langford. Beating the hold-out: Bounds for k-fold and progressive cross-validation. In *Proceedings of the twelfth annual conference on Computational learning theory*, pages 203–208, 1999.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965, 1989.

Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of machine learning research*, 2(Mar):499–526, 2002.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR, 2020.

Eugenio Clerico, Tyler Farghly, George Deligiannidis, Benjamin Guedj, and Arnaud Doucet. Generalisation under gradient descent via deterministic pac-bayes. *arXiv preprint arXiv:2209.02525*, 2022.

Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

Nan Ding, Xi Chen, Tomer Levinboim, Soravit Changpinyo, and Radu Soricut. Pactran: Pac-bayesian metrics for estimating the transferability of pretrained models to classification tasks. In *European Conference on Computer Vision*, pages 252–268. Springer, 2022.

Gintare Karolina Dziugaite and Daniel M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data, 2017. URL https://arxiv.org/abs/1703.11008.

Pascal Germain, Alexandre Lacasse, Francois Laviolette, Mario Marchand, and Jean-Francis Roy. Risk bounds for the majority vote: From a pac-bayesian analysis to a learning algorithm. *arXiv preprint arXiv:1503.08329*, 2015.

Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International conference on machine learning*, pages 1225–1234. PMLR, 2016.

Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993.

Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgIPJBFvH.

Michael Kearns and Dana Ron. Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. In *Proceedings of the tenth annual conference on Computational learning theory*, pages 152–162, 1997.

Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *European conference on computer vision*, pages 491–507. Springer, 2020.

Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2661–2671, 2019.

John Langford and Robert Schapire. Tutorial on practical prediction theory for classification. *Journal of machine learning research*, 6(3), 2005.

Sanae Lotfi, Marc Finzi, Sanyam Kapoor, Andres Potapczynski, Micah Goldblum, and Andrew Gordon Wilson. Pac-bayes compression bounds so tight that they can explain generalization, 2022. URL https://arxiv.org/abs/2211.13609.

David A McAllester. Some pac-bayesian theorems. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 230–234, 1998.

Vaishnavh Nagarajan and J Zico Kolter. Deterministic pac-bayesian generalization bounds for deep networks via generalizing noise-resilience. *arXiv preprint arXiv:1905.13344*, 2019.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on learning theory*, pages 1376–1401. PMLR, 2015.

Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1707.09564*, 2017.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

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

María Pérez-Ortiz, Omar Rivasplata, John Shawe-Taylor, and Csaba Szepesvári. Tighter risk certificates for neural networks, 2021. URL https://arxiv.org/abs/2007.12911.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? *CoRR*, abs/1902.10811, 2019. URL http://arxiv.org/abs/1902.10811.

Matthias Seeger. Pac-bayesian generalisation error bounds for gaussian process classification. *Journal of machine learning research*, 3(Oct):233–269, 2002.

Matthias Seeger, John Langford, and Nimrod Megiddo. An improved predictive accuracy bound for averaging classifiers. In *Proceedings of the 18th International Conference on Machine Learning*, pages 290–297, 2001.

Jamie Simon, Daniel Kunin, Alexander Atanasov, Enric Boix-Adserà, Blake Bordelon, Jeremy Cohen, Nikhil Ghosh, Florentin Guth, Arthur Jacot, Mason Kamb, Dhruva Karkada, Eric J. Michaud, Berkan Ottlik, and Joseph Turnbull. There will be a scientific theory of deep learning, 2026. URL https://arxiv.org/abs/2604.21691.

Khoat Than and Dat Phan. Non-vacuous generalization bounds for deep neural networks without any modification to the trained models, 2025. URL https://arxiv.org/abs/2503.07325.

Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

Guillermo Valle-Pérez and Ard A Louis. Generalization bounds for deep learning. *arXiv preprint arXiv:2012.04115*, 2020.

Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity: festschrift for alexey chervonenkis*, pages 11–30. Springer, 2015.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742, 2018.

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

Shuofeng Zhang and Ard Louis. Position: Many generalization measures for deep learning are fragile. *arXiv preprint arXiv:2510.18934*, 2025.

Wenda Zhou, Victor Veitch, Morgane Austern, Ryan P. Adams, and Peter Orbanz. Non-vacuous generalization bounds at the imagenet scale: A pac-bayesian compression approach, 2019. URL https://arxiv.org/abs/1804.05862.

A Appendix / supplemental material

B Related work

While the history of generalization theory spans VC and uniform-convergence analyses, PAC learning, Rademacher complexity, margin and norm-based bounds, algorithmic stability, PAC-Bayes, compression, and modern non-vacuous deep-network certificates [Vapnik and Chervonenkis, 2015, Bartlett and Mendelson, 2002, Valiant, 1984, Blumer et al., 1989, Bartlett et al., 1998, Bartlett, 2002, McAllester, 1998, Hardt et al., 2016, Neyshabur et al., 2015, Bartlett et al., 2017, Arora et al., 2018, Dziugaite and Roy, 2017], we focus on a smaller universe: computable numerical certificates for a fixed trained predictor. In that narrower sense, our proof template is closer to the classical hold-out tradition than to global capacity measures. Practical prediction theory has long emphasized that fresh-sample error, combined with a sharp concentration inequality, already yields useful guarantees [Langford and Schapire, 2005]. The same perspective extends naturally to K-fold and progressive validation [Blum et al., 1999], leave-one-out and algorithmic-stability analyses [Kearns and Ron, 1997, Bousquet and Elisseeff, 2002], and modern SGD-based stability bounds [Hardt et al., 2016]. Our cross-fit analysis is squarely in this lineage: the mathematics is intentionally elementary, relying on held-out Bernoulli errors, cross-fitting, and KL concentration. The unusual step is the object being evaluated—a simple k-NN rule in the learned representation [Cover and Hart, 1967].

The strongest computable non-vacuous certificates for deep networks have mostly come from PAC-Bayes and compression. Classical PAC-Bayes work established general-purpose risk bounds for randomized predictors [McAllester, 1998, Seeger et al., 2001, Seeger, 2002], and modern deep-learning instantiations include spectrally normalized or localized certificates with increasingly tight numerical performance [Neyshabur et al., 2017, Dziugaite and Roy, 2017, Pérez-Ortiz et al., 2021]. A parallel line certifies models through description length, pruning, or compression [Hinton and Van Camp, 1993, Arora et al., 2018, Zhou et al., 2019, Lotfi et al., 2022]. These methods are the natural numerical baselines for any new certificate, but they typically reason through parameter-space perturbation, posterior complexity, or compressibility. A positioning nuance important here is that PAC-Bayes is not only a parameter-space story: function-space and marginal-likelihood PAC-Bayes analyses also exist [Valle-Pérez and Louis, 2020, Zhang and Louis, 2025]. Our contribution is therefore not "the first non-parameter-space bound" in the broadest sense, but a particularly simple representation-space certificate based on held-out error, agreement, and stability.

A smaller recent literature has moved closer to certifying the exact deployed deterministic predictor. [Nagarajan and Kolter, 2019] derive deterministic PAC-Bayes bounds via noise resilience, and [Clerico et al., 2022] study deterministic PAC-Bayes bounds along gradient-descent trajectories. [Than and Phan, 2025] obtain non-vacuous bounds for unmodified trained networks through an a priori data partition rather than post-hoc stochastic perturbation or compression. Our results belong to this deterministic family, but the route is different: instead of certifying the predictor through weight perturbations, trajectory quantities, or partition structure alone, we certify a simple representation-space decoder and then bridge from decoder to the deployed network.

A particularly close modern comparator is disagreement-based certification. PAC-Bayesian majority-vote analyses already show that disagreement can be a certificate-relevant quantity [Germain et al., 2015], and [Bazinet et al., 2026] turn this into a general recipe: certify an auxiliary predictor, then use unlabeled disagreement to transfer the guarantee to the target model. Our method is spiritually close to that template, but with less auxiliary machinery. The surrogate is not separately trained; it is a parameter-free k-NN decoder induced by the representation itself. The unlabeled term is also more interpretable: it measures fold/full prediction stability, which reconnects the argument to the classical stability literature [Bousquet and Elisseeff, 2002, Hardt et al., 2016].

Outside the certification literature, the premise that a frozen representation should be judged by simple probes is standard in modern vision. Nearest-neighbor evaluation appears already in non-parametric instance discrimination [Wu et al., 2018] and remains common, alongside linear probing, in transfer and self-supervised learning [Kornblith et al., 2019, Kolesnikov et al., 2020, Chen et al., 2020, Caron et al., 2021, Oquab et al., 2023]. Relatedly, Ding et al. [2022] use PAC-Bayesian quantities to estimate the transferability of pretrained representations. Our contribution is therefore not the observation that k-NN is a useful probe. It is that, with cross-fitting, agreement, and stability, held-out probe performance becomes a formal certificate for the deployed network itself.

The empirical claims of this paper also touch two further literatures. First, modern work on double descent studies the interpolation peak in model size, training time, and sample size [Belkin et al., 2019, Nakkiran et al., 2021]. Second, neural-collapse work studies the simple class geometry that emerges in the last layer of trained classifiers, including nearest-class-center behavior [Papyan et al., 2020]. Our **Q1** term is not a neural-collapse theorem, but it measures a nearby object: local class geometry sufficient for a simple local classifier to succeed. Likewise, our causal manipulation of **Q1** should be read against memorization results showing that deep networks can fit labels without learning useful structure [Zhang et al., 2017]. Finally, unlike post-hoc complexity measures that are primarily judged by correlation with test error across model families [Jiang et al., 2020], our method returns a finite-sample upper bound.

C Further results

C.1 SiLU comparison

		MNIST (n=60k)			CIFAR-10 (n=50k)			CIFAR-100 (n=50k)		
Model	Act.	Test	Bound	Gap	Test	Bound	Gap	Test	Bound	Gap
Feedforward networks										
FFNN	ReLU	1.5	4.9$_{\pm 0.4}$	**3.4**	44.6	95.5$_{\pm 1.1}$	50.9	75.2	100.0	24.8
FFNN	SiLU	1.4	4.8$_{\pm 0.1}$	**3.4**	44.3	96.2$_{\pm 1.0}$	51.9	75.4	100.0	24.6
Convolutional networks										
ResNet-18	ReLU	0.6	**2.6**$_{\pm 0.0}$	**2.0**	4.7	13.0$_{\pm 0.3}$	8.3	21.2	51.1$_{\pm 0.1}$	29.9
ResNet-18	SiLU	0.7	2.7$_{\pm 0.1}$	**2.0**	5.5	15.4$_{\pm 0.1}$	9.9	23.0	58.6$_{\pm 0.2}$	35.6
ResNet-50	ReLU	0.7	2.9$_{\pm 0.2}$	**2.2**	**4.3**	**12.4**$_{\pm 0.4}$	8.0	**19.4**	**44.6**$_{\pm 0.3}$	25.2
ResNet-50	SiLU	0.9	3.4$_{\pm 0.1}$	**2.5**	5.3	14.3$_{\pm 0.2}$	8.9	21.8	55.9$_{\pm 0.5}$	34.0
Vision Transformers (from scratch)										
ViT	ReLU	0.9	3.5$_{\pm 0.2}$	**2.6**	12.3	31.9$_{\pm 0.5}$	19.6	37.1	99.3$_{\pm 0.2}$	62.3
ViT	SiLU	0.9	3.7$_{\pm 0.1}$	**2.8**	12.3	32.3$_{\pm 0.3}$	20.1	38.2	100.0	61.8

Table 3: **Cross-fit certification of trained networks** (ℓ_2, J=5, δ=δ'=0.025). Best bound over $k \in \{1, 3, 5, 7, 11, 15, 21\}$ and $j \in [5]$. Each entry: mean $\pm$ std over 3 seeds.

Table 4: Desiderata for generalization certificates. Our approach satisfies all desiderata; the stability term is certified empirically from unlabeled data rather than bounded analytically.

Method	Det. net	No mod. training	No surrogate	No compress.	No stoch. weights	Pretrained / transfer	Arch.-agnostic	Diagnoses repr.
Dziugaite and Roy [2017]	✗	✗	✓	✓	✗	✗	✓	✗
Pérez-Ortiz et al. [2021]	✗	✗	✓	✓	✗	✗	✓	✗
Zhou et al. [2019]	✗	✗	✓	✗	✗	✗	✓	✗
Lotfi et al. [2022]	✗	✗	✓	✗	✗	✓	✓	∼
Bazinet et al. [2026]	✓	✗	✗	∼	✓	✗	✓	✗
Than and Phan [2025]	✓	✓	✓	✓	✓	✓	✗	✗
Ours	✓	✓	✓	✓	✓	✓	✓	✓

D Proofs

D.1 Proof of Theorem 2.2

Proof. Fix $j \in [J]$ and $k \in \mathcal{K}$. Conditioned on the complement sample $\mathcal{S}_{-j}$, the fold-j parameters $\boldsymbol{\theta}^{(-j)} = \mathcal{A}(\mathcal{S}_{-j})$ are fixed, so the decoder $h_j^{(k)}$ is a deterministic function (the reference set and the k-NN rule are fully determined by $\mathcal{S}_{-j}$ and the pre-specified tie-breaking). The fold points $\{(\boldsymbol{x}_i, y_i)\}_{i \in F_j}$ are i.i.d. draws from P that are conditionally independent of $\mathcal{S}_{-j}$.

Define the conditional risk

$$R_j^{(k)} \;=\; \Pr_{(\boldsymbol{X},Y)\sim P}\big(h_j^{(k)}(\boldsymbol{X}) \neq Y \mid \mathcal{S}_{-j}\big).$$

The indicators $Z_i = \mathbf{1}\big[h_j^{(k)}(\boldsymbol{x}_i) \neq y_i\big]$, $i \in F_j$, are conditionally i.i.d. $\mathrm{Bernoulli}\big(R_j^{(k)}\big)$, and their empirical mean is $\widehat{R}_j^{(k)} = \frac{1}{n_j}\sum_{i\in F_j} Z_i$.

We apply the KL concentration inequality for i.i.d. Bernoulli random variables [see e.g. **?**, Theorem 1]: for n i.i.d. $\mathrm{Bernoulli}(p)$ variables with empirical mean $\hat{p}$,

$$\Pr\big(\mathrm{kl}(\hat{p}\|p) > c\big) \;\le\; (n+1)\, e^{-nc} \qquad \text{for all } c > 0.$$

Applied conditionally on $\mathcal{S}_{-j}$, with $\delta' > 0$ to be chosen, this gives

$$\Pr\big(R_j^{(k)} > \mathrm{kl}_+^{-1}\big(\widehat{R}_j^{(k)},\ \tfrac{\log((n_j+1)/\delta')}{n_j}\big) \;\big|\; \mathcal{S}_{-j}\big) \;\le\; \delta' \qquad \text{a.s.}$$

Define the event

$$E_{j,k} \;=\; \Big\{R_j^{(k)} \le \mathrm{kl}_+^{-1}\big(\widehat{R}_j^{(k)},\ \tfrac{\log((n_j+1)/\delta')}{n_j}\big)\Big\}.$$

The conditional bound gives $\Pr(E_{j,k}^c \mid \mathcal{S}_{-j}) \le \delta'$ a.s., so by the tower property:

$$\Pr(E_{j,k}^c) \;=\; \mathbb{E}\big[\Pr(E_{j,k}^c \mid \mathcal{S}_{-j})\big] \;\le\; \delta'.$$

A union bound over all $|\mathcal{K}| \cdot J$ pairs (k, j) gives

$$\Pr\Big(\exists\, j \in [J],\ k \in \mathcal{K} \;:\; E_{j,k}^c\Big) \;\le\; |\mathcal{K}| \cdot J \cdot \delta'.$$

Setting $\delta' = \delta/(|\mathcal{K}| \cdot J)$ yields: with probability at least $1-\delta$ over $\mathcal{S}_n$, for all $j \in [J]$ and $k \in \mathcal{K}$,

$$R_j^{(k)} \;\le\; \mathrm{kl}_+^{-1}\left(\widehat{R}_j^{(k)},\ \frac{\log\big(|\mathcal{K}|\, J/\delta\big) + \log(n_j+1)}{n_j}\right),$$

which is Equation (1). □

D.2 Proof of Theorem 2.3

Proof. Since φ is fixed by assumption, conditioning on $\mathcal{S}_{\mathrm{ref}}$ makes the decoder $h^{(k)}$ deterministic (the reference set fixes the k-NN rule). The evaluation points $\{(\boldsymbol{x}_i, y_i)\}_{i \in I_{\mathrm{eval}}}$ are therefore conditionally i.i.d. from P.

The error indicators $Z_i^{(k)} = \mathbf{1}\big[h^{(k)}(\boldsymbol{x}_i) \neq y_i\big]$, $i \in I_{\mathrm{eval}}$, are conditionally i.i.d. $\mathrm{Bernoulli}(R^{(k)})$, where $R^{(k)} = \Pr(\, h^{(k)}(\boldsymbol{X}) \neq Y \mid \mathcal{S}_{\mathrm{ref}})$. The KL concentration inequality applied conditionally on $\mathcal{S}_{\mathrm{ref}}$ gives, for each $k \in \mathcal{K}$:

$$\Pr\big(R^{(k)} > \mathrm{kl}_+^{-1}\big(\widehat{R}^{(k)},\ \tfrac{\log((n_e+1)/\delta')}{n_e}\big) \;\big|\; \mathcal{S}_{\mathrm{ref}}\big) \;\leq\; \delta' \quad \text{a.s.}$$

By the tower property, each conditional bound promotes to an unconditional one (as in the proof of Theorem 2.2). A union bound over all $|\mathcal{K}|$ values of k gives total failure probability at most $|\mathcal{K}| \cdot \delta'$. Setting $\delta' = \delta/|\mathcal{K}|$ yields

$$c_e \;=\; \frac{\log\big(|\mathcal{K}|/\delta\big) + \log(n_e{+}1)}{n_e},$$

which is (2). □

D.3 Proof of Theorem 2.4

Proof. The indicator decomposition

$$\mathbf{1}\big[\widehat{f}_j(\boldsymbol{x}) \neq y\big] \;\leq\; \mathbf{1}\big[h_j^{(k)}(\boldsymbol{x}) \neq y\big] \;+\; \mathbf{1}\big[h_j^{(k)}(\boldsymbol{x}) \neq \widehat{f}_j(\boldsymbol{x})\big]$$

splits the fold network's risk into the population decoder error and the population disagreement rate:

$$R_j^{(k)} = \Pr\big(h_j^{(k)}(\boldsymbol{X}) \neq Y \mid \mathcal{S}_{-j}\big), \qquad D_j^{(k)} = \Pr\big(h_j^{(k)}(\boldsymbol{X}) \neq \widehat{f}_j(\boldsymbol{X}) \mid \mathcal{S}_{-j}\big).$$

Conditioned on $\mathcal{S}_{-j}$, both $h_j^{(k)}$ and $\widehat{f}_j$ are deterministic, so the fold points $\{(\boldsymbol{x}_i, y_i)\}_{i \in F_j}$ are conditionally i.i.d. from P. This produces two families of conditionally i.i.d. Bernoulli indicators:

$$Z_i^{(k)} = \mathbf{1}\big[h_j^{(k)}(\boldsymbol{x}_i) \neq y_i\big] \sim \mathrm{Bernoulli}\big(R_j^{(k)}\big), \qquad W_i^{(k)} = \mathbf{1}\big[h_j^{(k)}(\boldsymbol{x}_i) \neq \widehat{f}_j(\boldsymbol{x}_i)\big] \sim \mathrm{Bernoulli}\big(D_j^{(k)}\big),$$

with empirical means $\widehat{R}_j^{(k)}$ and $\widehat{D}_j^{(k)}$ respectively.

There are $2 \cdot |\mathcal{K}| \cdot J$ events to control (two families $\times$ $|\mathcal{K}|$ values of $k \times J$ folds). Assigning confidence budget $\delta' = \delta/(2\,|\mathcal{K}|\,J)$ to each, the KL concentration inequality applied conditionally on $\mathcal{S}_{-j}$ gives

$$\Pr\big(R_j^{(k)} > \mathrm{kl}_+^{-1}\big(\widehat{R}_j^{(k)},\ \tfrac{\log((n_j+1)/\delta')}{n_j}\big) \;\big|\; \mathcal{S}_{-j}\big) \;\leq\; \delta' \quad \text{a.s.},$$

and identically for $D_j^{(k)}$ with $\widehat{D}_j^{(k)}$. By the tower property, each conditional bound promotes to an unconditional one. A union bound over all $2\,|\mathcal{K}|\,J$ events gives total failure probability at most δ. Substituting $\delta' = \delta/(2\,|\mathcal{K}|\,J)$ into the log yields $c_j = \big[\log(2\,|\mathcal{K}|\,J/\delta) + \log(n_j{+}1)\big]/n_j$, and the risk decomposition $\Pr(\widehat{f}_j(\boldsymbol{X}) \neq Y \mid \mathcal{S}_{-j}) \leq R_j^{(k)} + D_j^{(k)}$ gives (3). □

D.4 Proof of Theorem 2.5

Proof. Conditioned on $\mathcal{S}_n$, both $\widehat{f}_j$ and $\widehat{f}^{(\mathrm{full})}$ are deterministic. The indicators $V_r = \mathbf{1}[\widehat{f}_j(\boldsymbol{x}_r) \neq \widehat{f}^{(\mathrm{full})}(\boldsymbol{x}_r)]$, $r = 1, \ldots, m$, are therefore conditionally i.i.d. $\mathrm{Bernoulli}(\Delta_j^{\mathrm{stab}})$, with empirical mean $\widehat{\Delta}_j^{\mathcal{U}}$.

Assign confidence budget $\delta'' = \delta'/J$ to each fold. The KL concentration inequality applied conditionally on $\mathcal{S}_n$ gives

$$\Pr\big(\Delta_j^{\mathrm{stab}} > \mathrm{kl}_+^{-1}\big(\widehat{\Delta}_j^{\mathcal{U}},\ \tfrac{\log((m+1)/\delta'')}{m}\big) \;\big|\; \mathcal{S}_n\big) \;\leq\; \delta'' \quad \text{a.s.}$$

By the tower property, $\Pr(\cdot) = \mathbb{E}[\Pr(\cdot \mid \mathcal{S}_n)] \leq \delta''$. A union bound over $j \in [J]$ gives total failure probability at most $J \cdot \delta'' = \delta'$. Substituting $\delta'' = \delta'/J$ into the log yields the confidence radius $\big[\log(J/\delta') + \log(m{+}1)\big]/m$, which is (6). □

D.5 Proof of Theorem 2.6

Proof. The proof combines two high-probability events via a union bound.

Event 1 (probability $\geq 1-\delta$ over $\mathcal{S}_n$). Theorem 2.4 gives, for all $j \in [J]$ and $k \in \mathcal{K}$:

$$\Pr_{(\boldsymbol{X},Y)\sim P}\big(\widehat{f}_j(\boldsymbol{X}) \neq Y \mid \mathcal{S}_{-j}\big) \;\leq\; \mathrm{kl}_+^{-1}\big(\widehat{R}_j^{(k)}, c_j\big) + \mathrm{kl}_+^{-1}\big(\widehat{D}_j^{(k)}, c_j\big).$$

Event 2 (probability $\geq 1-\delta'$ over $(\mathcal{S}_n, \mathcal{U})$). Theorem 2.5 gives, for all $j \in [J]$:

$$\Delta_j^{\mathrm{stab}} = \Pr_{\boldsymbol{X}\sim P_X}\big(\widehat{f}_j(\boldsymbol{X}) \neq \widehat{f}^{(\mathrm{full})}(\boldsymbol{X}) \mid \mathcal{S}_n\big) \;\leq\; \mathrm{kl}_+^{-1}\big(\widehat{\Delta}_j^{\mathcal{U}}, c_m\big).$$

On the intersection of both events (probability $\geq 1-\delta-\delta'$), the indicator decomposition (4) gives

$$\Pr_{(\boldsymbol{X},Y)\sim P}\big(\widehat{f}^{(\mathrm{full})}(\boldsymbol{X}) \neq Y \mid \mathcal{S}_n\big) \;\leq\; \Pr_{(\boldsymbol{X},Y)\sim P}\big(\widehat{f}_j(\boldsymbol{X}) \neq Y \mid \mathcal{S}_n\big) \;+\; \Delta_j^{\mathrm{stab}}.$$

Since $\widehat{f}_j$ depends on $\mathcal{S}_n$ only through $\mathcal{S}_{-j}$, the first term equals $\Pr(\widehat{f}_j(\boldsymbol{X}) \neq Y \mid \mathcal{S}_{-j})$, which is bounded by Event 1. Substituting both bounds yields (7). □

E Experimental Details

This appendix collects all information needed to reproduce the experiments. Section E.1 describes shared ingredients; Sections E.3–E.10 give per-table details; Sections E.6–E.9 give per-figure details; Section E.11 records seeds and compute.

E.1 Common setup

Datasets. The supervised experiments use the standard train/test splits of MNIST, CIFAR-10, CIFAR-100, and SVHN. The transfer experiments additionally use the ImageNet ILSVRC-2012 validation set and ImageNet-V2. Evaluation is always deterministic and uses the standard held-out test or validation inputs.

Representation and decoder. The representation $\varphi(x)$ is the final hidden feature vector immediately before the linear readout: the output of global average pooling for convolutional networks, and the class-token embedding after the final layer norm for transformers. The decoder is a majority-vote k-nearest-neighbor rule in ℓ_2 representation space, searching over $k \in \{1, 3, 5, 7, 11, 15, 21\}$. When a single bound is reported it is always the minimum over this grid; the union bound over $|K| = 7$ values of k is included in the confidence term. Distances are computed in `float64`; neighbor ties are broken by lowest reference index, and majority-vote ties by lowest class label.

Architectures. Table 5 summarizes the four architecture families used in the trained-model experiments. The transfer experiments use pretrained ResNet-18, ResNet-50, WRN-50-2, ViT-B/16, ViT-L/16, DINOv2-B/14, DINOv2-L/14, and DINOv2-g/14, plus a randomly initialized ResNet-18 as a null control (see Appendix F for extraction details).

Table 5: Architecture details for trained-model experiments.

Family	Repr. dim	Notes
FFNN	D=1024	6 hidden layers, width 1024, no dropout. Activation (ReLU or SiLU) applied before readout. No data augmentation.
ResNet-18	D=512	CIFAR-style stem (3×3 conv, no maxpool) for 32×32 inputs. Standard torchvision stem for 224×224.
ResNet-50	D=2048	Same stem conventions as ResNet-18.
ViT (from scratch)	varies	MNIST: patch 4, embed 192, depth 4, 3 heads, MLP ratio 4, dropout 0. CIFAR-10/100: patch 4, embed 256, depth 6, 8 heads, MLP ratio 4, dropout 0.

Table 6: Optimization hyperparameters for the cross-fit experiments (Table 2). All schedules are cosine decay to zero unless stated otherwise. "LS" denotes label smoothing.

Family	Dataset	Optimizer	LR	Momentum	WD	LS	Batch	Epochs
FFNN	MNIST	AdamW	10^{-3}	–	10^{-5}	0	512	150
	CIFAR-10	AdamW	$5{\cdot}10^{-4}$	–	$2{\cdot}10^{-5}$	0	512	300
	CIFAR-100	AdamW	$3{\cdot}10^{-4}$	–	$2{\cdot}10^{-5}$	0	512	400
ResNet-18	MNIST	SGD	0.05	0.9	10^{-4}	0	256	120
	CIFAR-10	SGD	0.1	0.9	$5{\cdot}10^{-4}$	0	128	200
	CIFAR-100	SGD	0.1	0.9	$5{\cdot}10^{-4}$	0.1	128	240
ResNet-50	MNIST	SGD	0.05	0.9	10^{-4}	0	128	120
	CIFAR-10	SGD	0.1	0.9	$5{\cdot}10^{-4}$	0	128	240
	CIFAR-100	SGD	0.1	0.9	$5{\cdot}10^{-4}$	0.1	128	300
ViT	MNIST	AdamW	$5{\cdot}10^{-4}$	–	0.01	0	256	200
	CIFAR-10	AdamW	$5{\cdot}10^{-4}$	–	0.05	0.1	128	300
	CIFAR-100	AdamW	$5{\cdot}10^{-4}$	–	0.05	0.1	128	300

ViT uses a 10-epoch linear warmup before cosine decay. CIFAR-scale convnets and ViTs use standard random-crop-and-flip augmentation during training; FFNNs use no augmentation. All AdamW runs use default $(\beta_1, \beta_2) = (0.9, 0.999)$.

Optimization recipes. Table 6 lists every hyperparameter for the cross-fit experiments of Table 2. Within each architecture family the recipe is fixed across activations; differences across datasets are shown explicitly.

E.2 Cross-fit construction

The cross-fit procedure is shared by Table 2, Figure 1b, and Figure 2. We describe it once here; later subsections refer back.

Fold partition. The n training indices are permuted with a fixed fold seed (42) and divided into $J{=}5$ contiguous blocks. The partition is a single uniform random permutation; it is *not* class-stratified. For the balanced datasets used here, a five-way random split still yields large per-class counts in every fold (about 1 200 per class for MNIST, 1 000 for CIFAR-10, 100 for CIFAR-100), so we accept the realized folds without imposing balance.

Fold-model training. Each fold model is trained on the $0.8n$ complement examples using the *same* architecture, hyperparameters, and number of epochs as the full-data model. The batch size is unchanged, so a fold model executes fewer gradient steps per epoch than the full model. We intentionally do not compensate for this by rescaling epochs or learning rates: the Q3 term is designed to measure the stability of the *entire training recipe* under fold removal, including any optimization-trajectory change induced by the smaller sample.

Certification computation. For each fold j and each k, we compute three quantities on the held-out fold and the independent unlabeled test set:

1. **Q1** (decoder error): fraction of held-out fold examples where the k-NN decoder disagrees with the true label.
2. **Q2** (decoder–network disagreement): fraction of the same held-out examples where the decoder disagrees with the fold-trained network.
3. **Q3** (fold–full stability): fraction of independent test inputs where the fold-trained and full-data networks disagree. Labels are not used.

Each empirical rate is converted to a KL upper bound with confidence radii

$$c_j = \frac{\log(2|K|\,J/\delta) + \log(|F_j|{+}1)}{|F_j|}, \qquad c_m = \frac{\log(J/\delta') + \log(m{+}1)}{m},$$

using $\delta{=}\delta'{=}0.025$. The three bounds are summed, and the reported certificate is $\min_{j,k}$ of that sum. The optimization over (j, k) is therefore part of the stated certificate, not an unaccounted model-selection step.

E.3 Table 1: Transfer certification

No network is trained on the target task. For each target dataset (CIFAR-10, CIFAR-100, ImageNet-val, ImageNet-V2), the labeled sample is split 80/20 into reference and evaluation portions using a fixed split seed 42. The k-NN decoder is built on the reference portion and evaluated on the held-out portion. Because the representation is fixed independently of the target data, Q2 and Q3 vanish by construction and the certificate reduces to a single KL bound on the decoder error.

The pretrained backbones are listed in Table 1 of the main text. For each backbone, the representation is extracted with a single deterministic forward pass using the preprocessing pipeline bundled with the official weights (see Appendix F).

E.4 Table 2: Cross-fit certification of trained networks

This table evaluates the full three-term certificate described in Section E.2, using the hyperparameters in Table 6 and the architectures in Table 5. Each configuration is run over multiple training seeds; the archived runs use seeds $\{0, 1, 2\}$ for MNIST and CIFAR-10 and seeds $\{0, 1\}$ for CIFAR-100. The table reports mean $\pm$ std over available seeds.

E.5 Table 3: Comparison with prior bounds

This table is a positioning summary, not a unified rerun. Baseline entries are copied from the corresponding publications; the "Certified object" column records what each method actually certifies (stochastic net, compressed net, surrogate, or deterministic net).

Source of "Ours" rows. All three "Ours" rows are cross-fit results, not transfer results:

- **MNIST:** ResNet-18 / ReLU, seeds $\{0, 1, 2\}$.
- **CIFAR-10:** ResNet-50 / ReLU, seeds $\{0, 1, 2\}$.
- **CIFAR-100:** ResNet-50 / ReLU, seeds $\{0, 1\}$.

E.6 Figure 1b: Sample-size sweep

A ReLU ResNet-18 is trained from scratch on CIFAR-10 subsets of size $n \in \{1\,000, 5\,000, 10\,000, 20\,000, 30\,000, 40\,000, 50\,000\}$. Each subset is sampled uniformly at random (not stratified). The training recipe is fixed across all subset sizes: SGD, lr 0.1, momentum 0.9, weight decay $5{\cdot}10^{-4}$, cosine decay, batch size 128, 200 epochs, standard crop-and-flip augmentation. The full cross-fit certificate (Section E.2) is evaluated at each n. Seeds: $\{0, 1, 2\}$ (training), 42 (fold partition).

E.7 Figure 2: Width sweep and double descent

Width-scaled ResNet-18 models are trained on CIFAR-10 at multipliers $\{1, 2, 2.5, 3, 3.5, 4, 4.5, 5, 6, 8, 10, 12, 16, 20, 24, 32\}$. (Exploratory runs at $48\times$ and $64\times$ were also trained but are omitted from the figure.) The training recipe is held fixed: SGD, lr 0.1, momentum 0.9, cosine decay, batch size 128, 400 epochs. Label noise is injected at levels 0%, 10%, and 20% by replacing each selected label with a uniformly random class. Seeds: $\{42, 43, 44\}$.

For each width, the plotted Q1 is the test-set decoder error at the k minimizing Q1 over the grid; Q2 is the decoder–network disagreement at that same k. The figure therefore decomposes test-error into a monotone representation term (Q1) and a readout-agreement term (Q2) that peaks near the interpolation threshold.

E.8 Figure 3: Causal geometry intervention

Architecture and training. Six-layer FFNN, hidden width **1024**, trained on SVHN with AdamW (lr 10^{-4}, weight decay 10^{-5}), cosine decay, batch size 1024, 5 000 epochs, seed 0. Four activation functions are shown: ReLU, SiLU, GELU, tanh.

Regularizer schedule. The representation-repulsion regularizer $\mathcal{L}_{\text{rep}}$ (Eq. 8 of the main text) uses a fixed strength λ=2500, identical across all four activations with no per-activation tuning. The coefficient is held at zero through epoch 5, then increased linearly to its full value over the next 500 epochs (i.e. fully active after epoch 505). The ramp allows the network to begin fitting the classification task before the repulsion term dominates.

E.9 Figure 4: Weight-noise diagnostic

Base model. This diagnostic uses a *separate* CIFAR-10 ResNet-18 checkpoint, distinct from the cosine-decay checkpoints used in the certification tables. It is trained with seed 42, SGD, lr 0.1, momentum 0.9, weight decay $5{\cdot}10^{-4}$, batch size 128, and a **multistep** schedule with drops by factor 0.2 at epochs 60, 120, and 160. A different schedule was chosen for this diagnostic because the experiment predates the unified cosine-decay recipe adopted for the rest of the paper; the qualitative conclusion (representation degrades more slowly than the readout) is not schedule-dependent.

Noise protocol. Isotropic Gaussian noise $\mathcal{N}(0, \sigma^2)$ is added independently to every trainable parameter. The noise levels are $\sigma \in \{10^{-4}, 3{\cdot}10^{-4}, 10^{-3}, 0.002, 0.003, 0.005, 0.008, 0.01, 0.015, 0.02, 0.03, 0.05, 0.1, 0.2, 0.3, 0.5, 1.0\}$. For each σ the noisy model is sampled **10 times**; both network accuracy and k-NN decoder accuracy (k=5, fixed 40 000/10 000 reference/query split of the CIFAR-10 training set) are averaged over these 10 draws. The second panel fixes σ=0.01 and varies activation function and label-corruption level.

This figure is purely diagnostic and does not contribute to any formal certificate.

E.10 Table 4: Desiderata comparison

Table 4 is qualitative. It records structural properties of each certification approach (deterministic vs. stochastic, modified training, compression, surrogates, transfer applicability, architecture-agnostic proof, diagnostic interpretability). No experimental detail is needed beyond what is stated in the table itself.

E.11 Seeds and compute

Table 7 collects all random seeds.

Table 7: Random seeds used in each experiment.

Experiment	Training seeds	Fold / split seed
Table 1 (transfer)	–	42
Table 2 (MNIST, CIFAR-10)	$\{0, 1, 2\}$	42
Table 2 (CIFAR-100)	$\{0, 1\}$	42
Table 3 (“Ours”)	same as Table 2	42
Figure 1b (sample-size sweep)	$\{0, 1, 2\}$	42
Figure 2 (width sweep)	$\{42, 43, 44\}$	42
Figure 3 (causal intervention)	$\{0\}$	–
Figure 4 (weight noise)	$\{42\}$	–

Compute. All runs are single-GPU, single-model jobs. Archived wall-clock metadata covers the following: Figure 1b: ≈ 10.8 GPU-hours (21 runs); Table 2: ≈ 102.7 GPU-hours (64 runs); Figure 2: ≈ 72.9 GPU-hours (162 runs); total logged: ≈ 186 GPU-hours. GPU model names were not recorded in the saved artifacts. Comparable timing metadata were not logged for the transfer, regularizer, and weight-noise experiments.

F Representation extraction details

All representations are extracted by a single deterministic forward pass with no modification to the network. No gradient is computed. Evaluation inputs use deterministic preprocessing only (resize, center crop, channel normalization); no stochastic augmentation is applied at extraction time.

Feedforward networks (FFNN). Our FFNNs consist of L fully connected hidden layers followed by a linear classification head. The representation $h_{\boldsymbol{\theta}}(x)$ is the post-activation output of the final hidden layer, i.e. the D-dimensional vector that is fed into the softmax classifier. For the experiments in Section 3.2 we use $D = 1024$ with $L = 6$ hidden layers. The activation function (ReLU or SiLU) is applied before readout; this is the same activation used throughout the network.

ResNets (ResNet-18, ResNet-50, WRN-50-2). The representation is the output of the global average pooling layer, flattened to a vector of dimension D. For ResNet-18, $D = 512$; for ResNet-50 and WRN-50-2, $D = 2048$. Concretely, we register a forward hook on `model.avgpool` and flatten the resulting tensor to shape (B, D). The final `fc` layer is not part of the representation.

Vision Transformers (ViT-B/16, ViT-L/16). For torchvision ViTs, the representation is the class token (`[CLS]`) extracted after the final encoder layer norm (`model.encoder.ln`). This yields $D = 768$ for ViT-B/16 and $D = 1024$ for ViT-L/16. We register a forward hook on the layer norm and slice the first token: `output[:, 0, :]`.

DINOv2 (DINOv2-B/14, DINOv2-L/14, DINOv2-g/14). DINOv2 models are loaded via `torch.hub` from the `facebookresearch/dinov2` repository. The forward pass returns the class-token embedding directly, so no hook is needed. Dimensions are $D = 768$ (B/14), $D = 1024$ (L/14), and $D = 1536$ (g/14). All DINOv2 backbones are used frozen with no fine-tuning.

Preprocessing. All models use ImageNet-normalized inputs ($\mu = (0.485, 0.456, 0.406)$, $\sigma = (0.229, 0.224, 0.225)$). For pretrained torchvision models, we use the preprocessing pipeline bundled with the official weights (`weights.transforms()`), which applies a deterministic resize and center crop to 224×224. For DINOv2 and randomly initialized models, we apply `Resize(256)`, `CenterCrop(224)`, `ToTensor()`, and the same normalization. For CIFAR experiments, models trained from scratch use 32×32 inputs with per-dataset normalization; stochastic augmentation (random crop, horizontal flip) is applied during training only and never during feature extraction or bound evaluation.

Numerical precision. All ℓ_2 distances for k-NN are computed in `float64`. Tie-breaking is deterministic: for the transfer experiments, a lexicographic perturbation of $10^{-12} \times$ (reference index) is added to squared distances, so that among equidistant neighbors the one with the smallest index wins. For the cross-fit experiments, ties are broken by stable sort order over reference indices, which produces the same lowest-index-wins rule. Majority-vote ties are broken in favor of the lowest class label (`argmax` over a vote-count tensor).

Computational cost. Feature extraction requires a single forward pass per input per model. The dominant cost is k-NN search, which is $O(n_{\text{ref}} \cdot n_{\text{query}} \cdot D)$ per value of k. For the largest transfer experiment (ImageNet, $n_{\text{ref}} = 40{,}000$, $n_{\text{eval}} = 10{,}000$, $D = 1536$), this takes approximately 30 seconds on a single A100 GPU. All k values share the same distance matrix, so the total cost scales linearly in $|\mathcal{K}|$ only through the top-k selection step.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims are stated in Section 1 and then matched to the formal certificate in Section 2 and Theorem 2.6 and the empirical evaluations in Sections 3.1, 3.2 and 4; assumptions and caveats are discussed in Section 5.

   Guidelines: The abstract and introduction state the paper's three-term representation-level certificate, distinguish the transfer and cross-fit settings, and summarize the empirical claims at the same scope supported by Section 2 and Sections 3–4. Scope, assumptions, and caveats are discussed in Section 5.

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The main limitations are discussed in Section 5, including the empirical certification of Q3, the focus on classification error, dependence on the chosen representation layer and metric, and the computational cost of cross-fit training.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The formal setup and assumptions are stated in Section 2.1, the main results are given in Theorems 2.2 to 2.4 and 2.6, and complete proofs are provided in Sections D.1 to D.5.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The main experimental procedures are specified in Sections 3.1 and 3.2, and detailed reproducibility information is provided in Sections E and F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Anonimized code to reproduce the experiments in this paper will be provided upon request during the review phase and openly released with the camera-ready version.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Datasets, splits, preprocessing, architectures, optimisation settings, fold construction, search grids, and per-experiment protocols are specified in Sections 3.1, 3.2, E.1 to E.4, E.6 to E.9 and F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Deterministic evaluations and repeated-run results are distinguished in Sections E.3, E.4, E.7 and E.9; the corresponding captions and appendix text state the number of runs or draws and whether values are reported as mean $\pm$ standard deviation.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Compute, seeds, and logged run times are reported in Section E.11, and the computational cost of representation extraction and k-NN search is further discussed in Section F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The experimental scope described in Sections 3.1, 3.2 and E uses standard public vision benchmarks and pretrained backbones and does not involve crowdsourcing or human-subject experimentation; the work is presented as foundational research rather than a deployed decision system.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper includes a broader-impacts discussion covering both positive uses of tighter and more interpretable generalization certificates and potential risks, including over-trust in certificates or overstatement of deployment readiness. The paper also clarifies that the work is foundational rather than a deployed decision system.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a high-risk generative model, scraped dataset, or other dual-use asset that would require special release safeguards. The work studies generalization certification for image classifiers on standard benchmarks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The datasets, pretrained models, and software assets used in the experiments are all openly published version and are all well-cited.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: This work does not release a new dataset, model, or code asset.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.