Out-of-Distribution Detection and Neural Collapse Analysis

Diego Fleury Corrêa de Moraes, Sergio Magalhães Contente

February 19, 2026

Abstract

This report presents an implementation and analysis of Out-of-Distribution (OOD) detection methods and the Neural Collapse phenomenon. We train a ResNet-18 classifier on CIFAR-100 and evaluate multiple OOD scoring techniques including MSP, Max Logit, Energy Score, Mahalanobis Distance, ViM, and NECO [1]. Additionally, we study Neural Collapse properties NC1-NC5 across different network layers.

1 Introduction

Out-of-Distribution detection is critical for deploying deep learning models in real-world applications. This work explores various OOD scoring methods and their connection to the Neural Collapse phenomenon observed in well-trained neural networks.

1.1 Objectives

- Train ResNet-18 on CIFAR-100
- Implement and compare 6 OOD detection methods
- Analyze Neural Collapse properties NC1-NC5
- Study layer-wise collapse behavior

2 Questions

2.1 Notebook 1 : Training the Model

(Located under *\notebooks\01_training.ipynb*)

This notebook is dedicated to the training procedure. The environment in which it took place was **google colab pro**, with a H100 GPU ($\approx$ 1h). We trained the model on the CIFAR100 dataset :

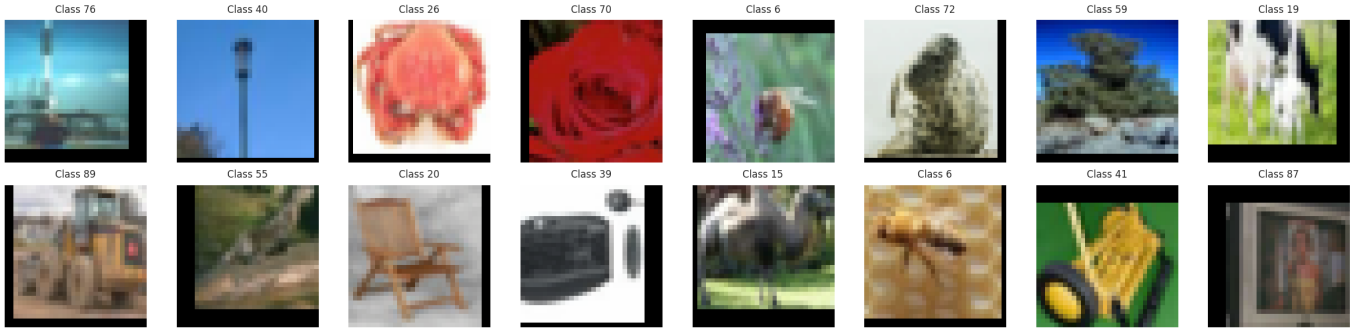


Figure 1: Sample batch from CIFAR100

2.1.1 Model Architecture

2.1.2 Model Architecture

(Located under *\src\utils\RF_search.py* and *\src\models\resnet.py*)

The book's ResNet18 model defines a first layer block (denoted 'b1') as being different from the rest, with a sequential call to a convolutional layer with 1 input channel, 64 output channels, 7x7 convolutions with stride=2 and padding=3, followed by batchnorm and maxpool (k=3, stride=2, padding=1). This neural net however was constructed with the ImageNet dataset in mind, which has 224x224 pixels of input resolution. In order to adapt it to the Cifar100 dataset, we run a small hyperparameter search (over the parameters for the 'b1' layer) with the goal of finding a receptive field ratio after the bottleneck stage (just before the model's 'neck') between 24 (75% of the total 32 pixels) and 160 (5×32) pixels. The following results are shown here :

```
TARGET: RF in [24, 160]px, spatial >= 2x2
======================================================================
k=3 s=2 p=1 pool      -> RF=431px (1x1) (b1_ratio = 0.21875) [BAD]
k=5 s=2 p=2 pool      -> RF=433px (1x1) (b1_ratio = 0.28125) [BAD]
```

```
k=7 s=2 p=3 pool     -> RF=435px (1x1) (b1_ratio = 0.34375) [BAD]
k=3 s=1 p=1 no_pool  -> RF=109px (4x4) (b1_ratio = 0.09375) [GOOD]
k=5 s=1 p=2 no_pool  -> RF=111px (4x4) (b1_ratio = 0.15625) [GOOD]
k=7 s=1 p=3 no_pool  -> RF=113px (4x4) (b1_ratio = 0.21875) [GOOD]
k=3 s=2 p=1 no_pool  -> RF=215px (2x2) (b1_ratio = 0.09375) [PARTIAL]
k=3 s=1 p=1 pool     -> RF=217px (2x2) (b1_ratio = 0.15625) [PARTIAL]
k=5 s=2 p=2 no_pool  -> RF=217px (2x2) (b1_ratio = 0.15625) [PARTIAL]
k=5 s=1 p=2 pool     -> RF=219px (2x2) (b1_ratio = 0.21875) [PARTIAL]
k=7 s=2 p=3 no_pool  -> RF=219px (2x2) (b1_ratio = 0.21875) [PARTIAL]
k=7 s=1 p=3 pool     -> RF=221px (2x2) (b1_ratio = 0.28125) [PARTIAL]

======================================================================
BEST CONFIG:
======================================================================
kernel=3, stride=1, padding=1, maxpool=False
Final RF: 109px, Spatial: 4x4
======================================================================
Best receptive field ratio (after just b1)= 9.38 %
======================================================================
```

It's interesting to see that the receptive field ratio after the 'b1' layer in the original setup (for ImageNet) was $\approx 4.91\%$ $((7 + (3 - 1) \times 2)/224)$, which is somewhat close to what we got here for the best value $(3/32 \approx 9.38\%)$ so, by both standards, we believe that this combination should yield a decent enough adaptation.

2.1.3 Model Training

Q1

Train a ResNet-18 classifier on CIFAR-100 using PyTorch.

In order to train the neural network, we begin by defining the basic **ResNet blocks**, in terms of segments composed of (2) **residual blocks**. The code for that is somewhat similar in function to that of the material, but with the slight architectural changes previously described. We also created some other files, for which the entire training process relies upon.

- *\src\data\datasets.py* : here, we define a couple of useful dataloader utilities : one for loading the main dataset (*get_cifar100_loaders*) and another for the other datasets, used for testing for OOD data (*get_ood_loaders*). Special care is taken their normalization as it should be consistent between datasets. Data augmentation (random cropping and horizontal flips) are also used for the CIFAR100 dataset.
- *\src\utils\training.py* : here, we define a Trainer class, that's to be used for optimization, validation-test accuracy monitoring, and loading-saving of checkpoints (for incremental training of the model).
- *\src\utils\visualization.py* : Similarly, a set of visualization functions (plots) were added for saving some useful figures and gifs.

Training used standard practices :

- **Optimizer** : Cossine Annealing scheduling, beginning at $lr = 0.1$, with weight decay and momentum updates
- **Loss function** : Standard (needed, in order to observe the Neural Collapse phenomenon) CrossEntropyLoss
- **Epochs & Checkpoints** : Total of 300 epochs, with checkpoints every 25 epochs in order to make sure that we can analyze the training procedure across time (*specially important for questions 4-6*).
- **Validation** : 10% of the training set.
- **Initialization** : Kaiming normal (fan-out mode) for CNNs, constant for weights (1) and biases (0) of batchnorm and linear layer's bias (0), with centered gaussian with small (0.01) variance for linear layer's weights

The training curves are as follows :

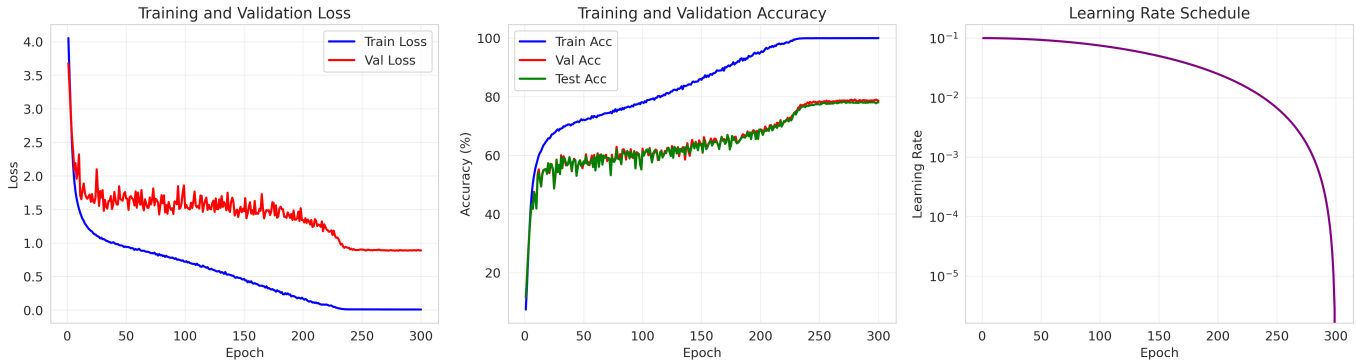


Figure 2: Training curves

We can observe that **terminal phase of training (TPT)** seemed to have occurred beginning about the 225th epoch, which means that we have 4 sample model parameter weights from this period (from epochs 225, 250, 275 and 300) to work on the analyses needed for questions 3 & 4.

2.2 Notebook 2 : OOD Detection Methods Scoring

For this section, we decided to separate the proposed scoring functions according to their methodologies :

- **In-Distribution**: CIFAR-100
- **Out-of-Distribution**: SVHN, CIFAR-10, Textures

Output-based Scores

(Located under \src\ood_scores\output_based)

- **MSP**: Maximum Softmax Probability
- **Max Logit**: Maximum logit value
- **Energy Score**: $E(x) = -\log \sum_c e^{f_c(x)}$

Distance-based Scores

(Located under \src\ood_scores\distance_based)

- **Mahalanobis**: Minimum Mahalanobis distance to class means

Feature-based Scores

(Located under \src\ood_scores\feature_based)

- **ViM**: Virtual Logit Matching
- **NECO**: Neural Collapse-based detection

Q2 & Q5

Implement and compare the OOD scores.

In this section, we checked and compared each of the OOD scoring methods. We chose, for organization purposes, to collect question 5 in this part as well, in order to provide a better comparison with the other methods. The chosen datasets for ood testing were SVHN, CIFAR10 and Textures, choosing (in a separate ood dataloader) 10% of their data for scoring (in order to get decent runtimes).

```
ID test samples: 10000
OOD samples:
(SVHN) : 2603
(CIFAR10) : 1000
(Textures) : 188
```

In order to properly incorporate the NECO method here, we opted to introduce a terminal phase of training mask, that skips evaluation of the loop for epochs in which we observed empirically that it hasn't been attained yet. During the evaluation loop, we standardized each method to output higher score for OOD samples, and lower ones for ID samples. The procedure took into account the evolution over the (saved) epochs of the classification, keeping track of both FPR95 and AUCROC metrics.

The figures show the evolution (terminal phase of training is shown in red), across datasets (and epochs) of the metrics, for all methods :

Output based methods :

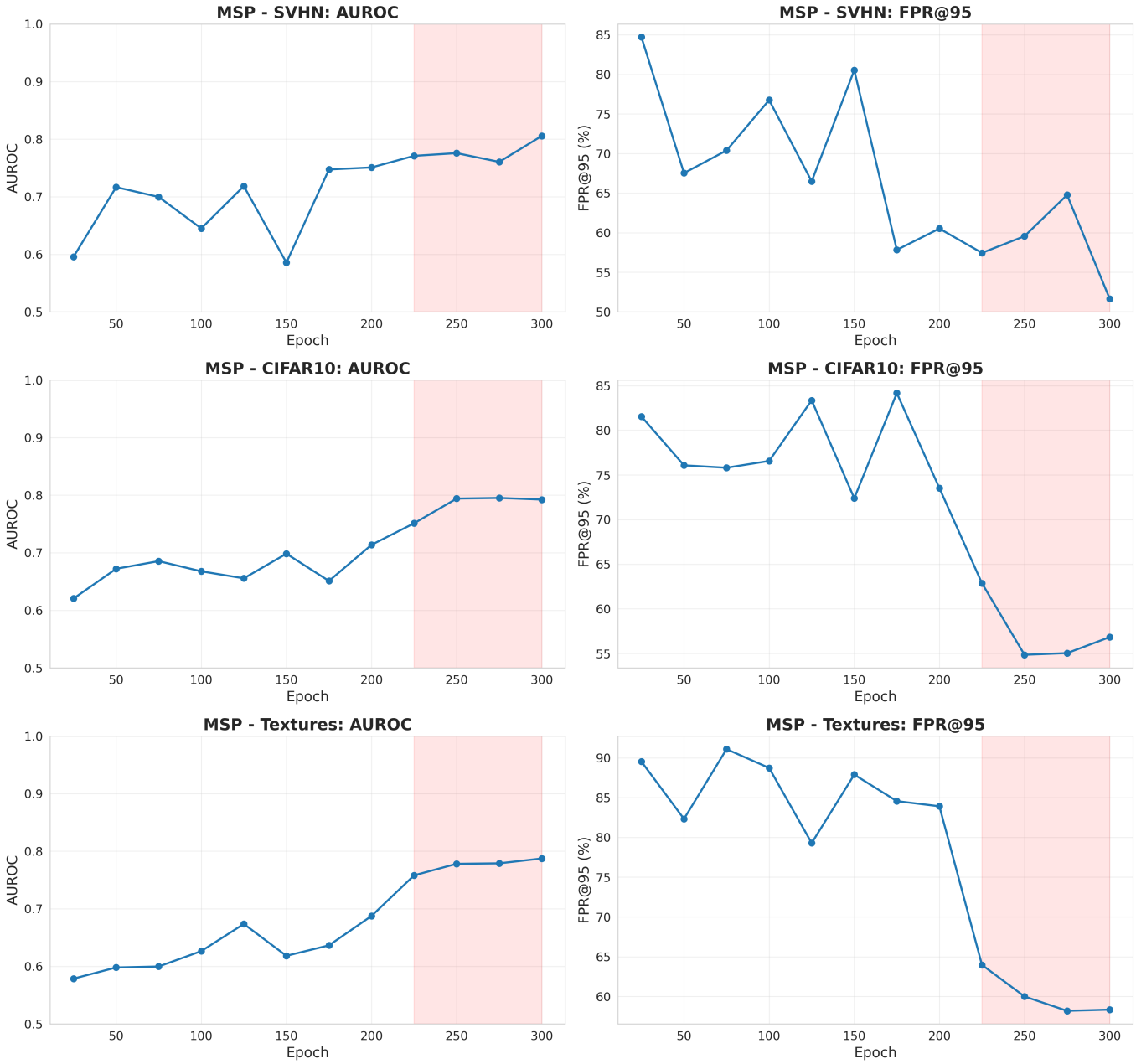


Figure 3: MSP method

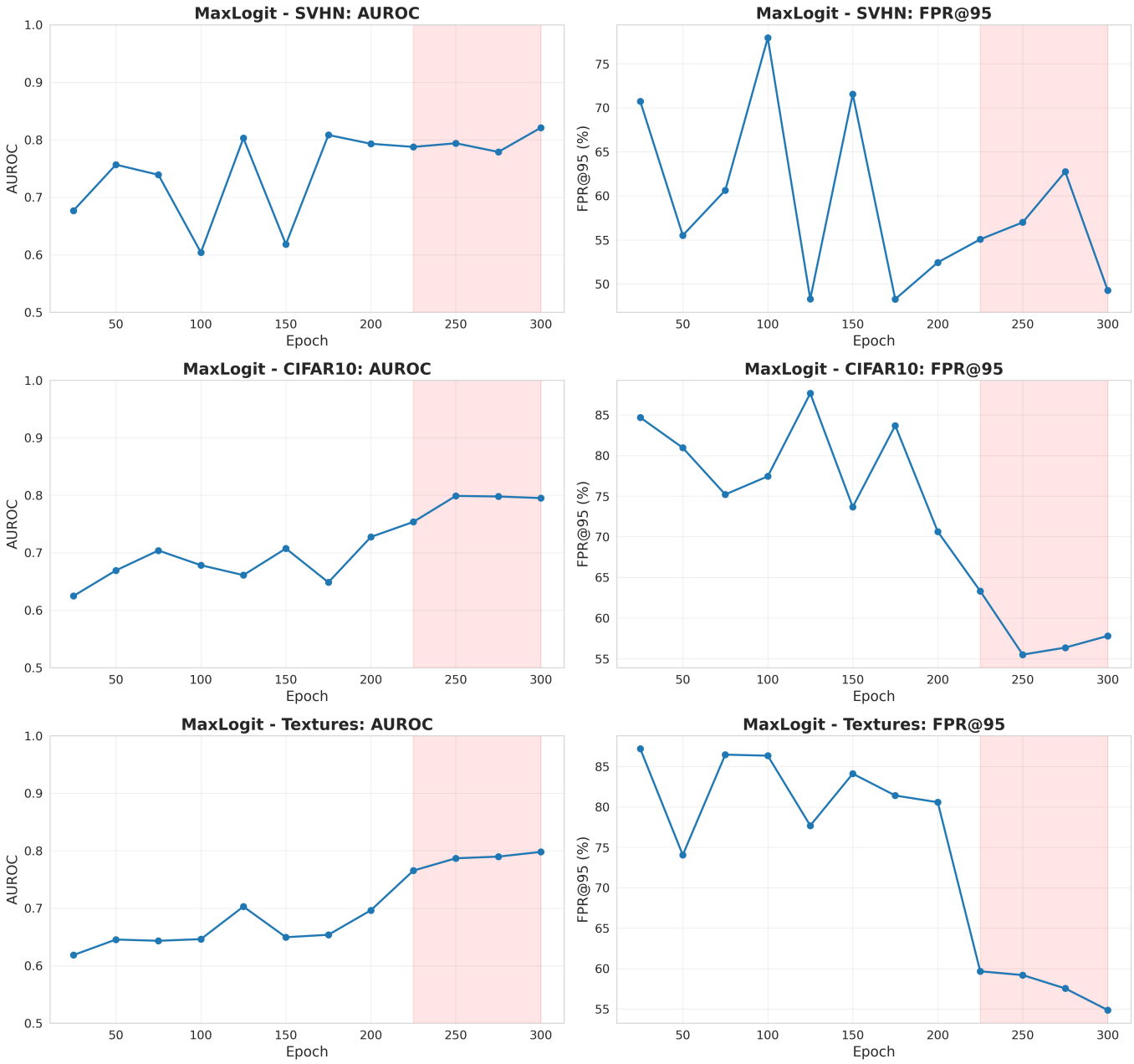


Figure 4: Max Logit method

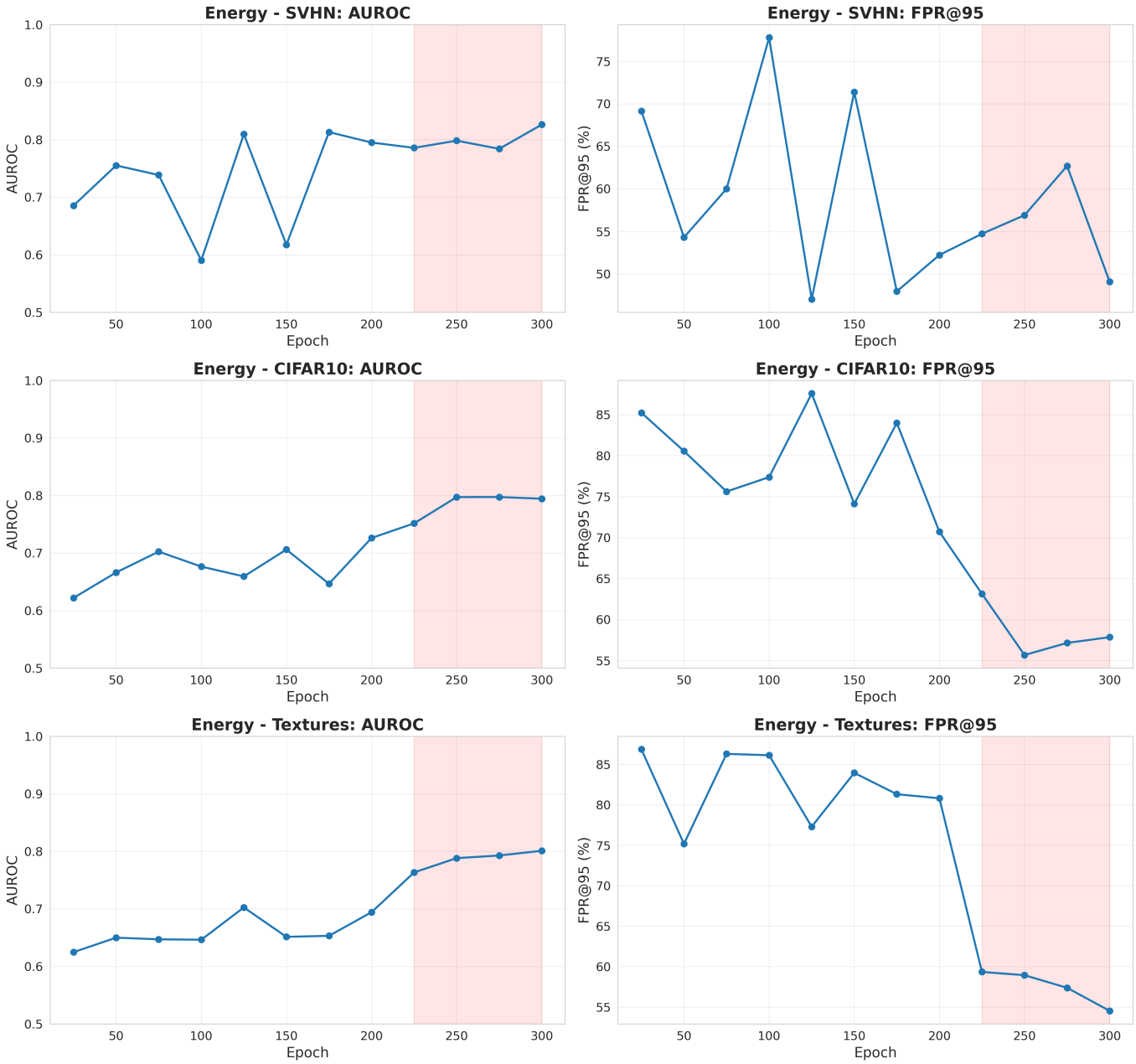


Figure 5: Energy method

Distance-based :

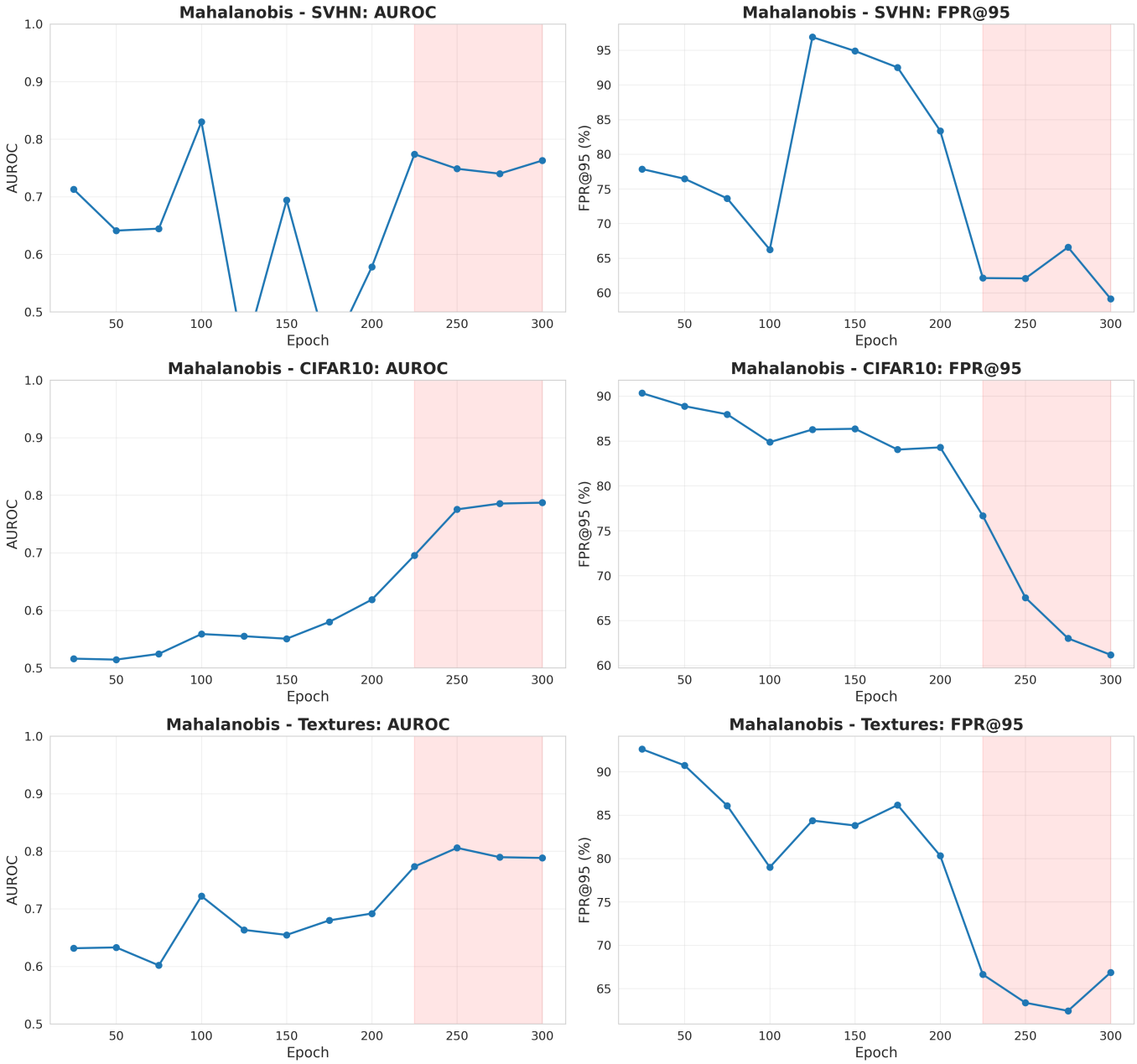


Figure 6: Mahalanobis method

Feature-based:

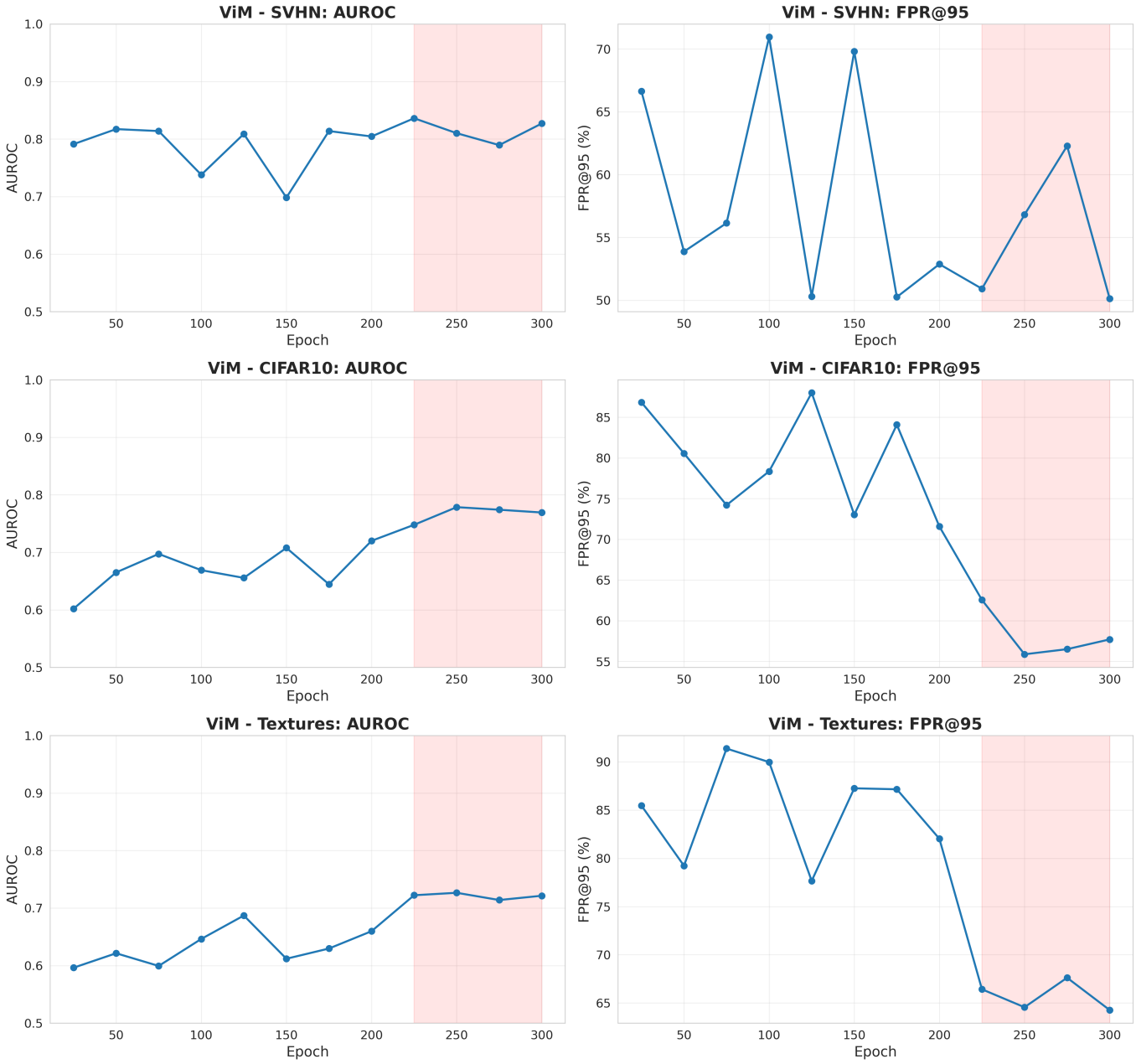


Figure 7: ViM method

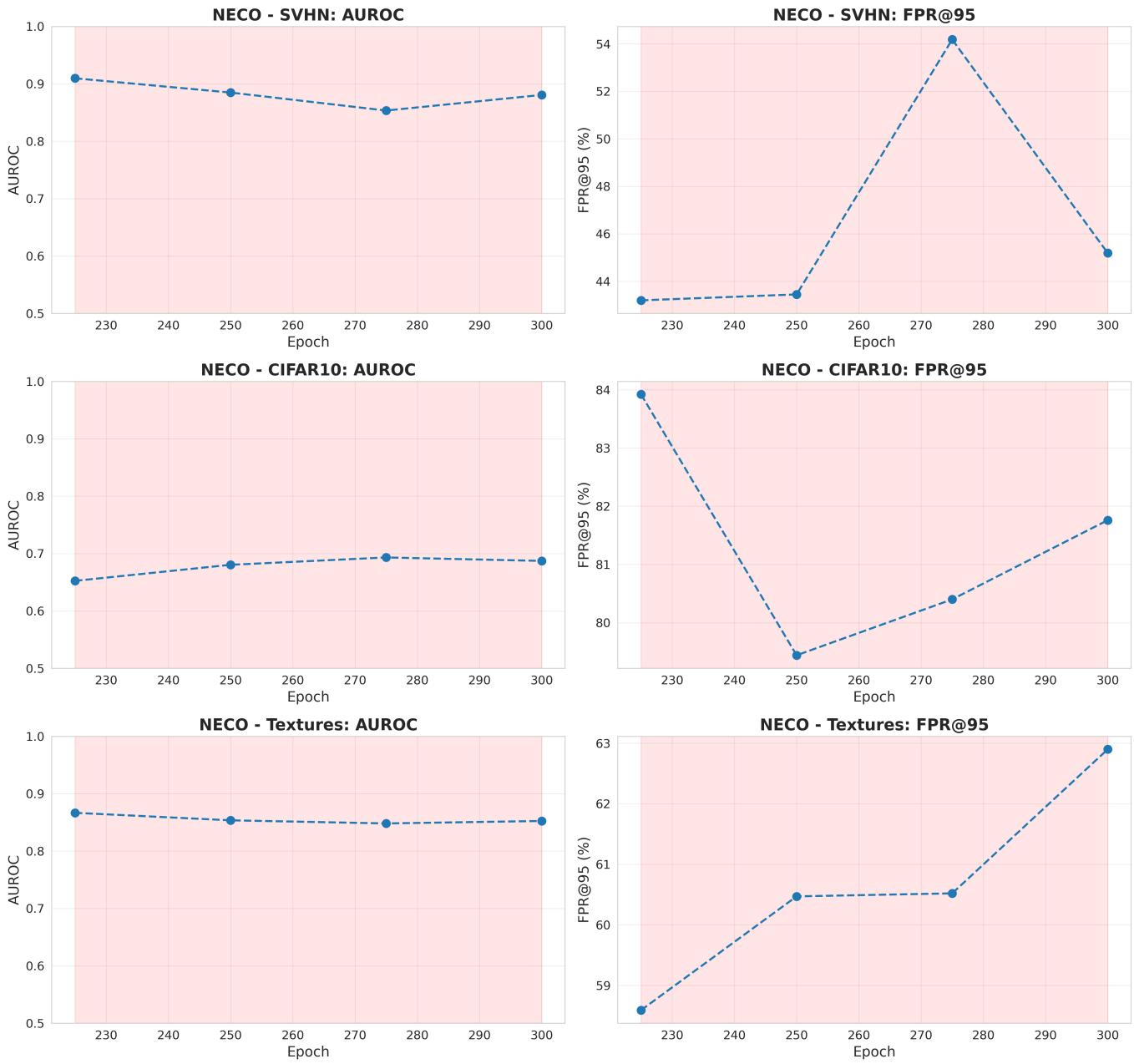


Figure 8: NECO method

The following table show the final results, at epoch 300.

Table 1: OOD Detection Results (AUROC ↑ / FPR95 ↓)

Method	Datasets					
	SVHN		CIFAR-10		Textures	
	AUROC	FPR95	AUROC	FPR95	AUROC	FPR95
MSP	0.81	0.52	0.79	**0.57**	0.79	0.58
Max Logit	0.82	0.49	**0.80**	0.58	0.80	0.55
Energy	0.83	0.49	0.79	0.58	0.80	**0.54**
Mahalanobis	0.76	0.59	0.79	0.61	0.79	0.67
ViM	0.83	0.50	0.77	0.58	0.72	0.64
NECO	**0.88**	**0.45**	0.69	0.82	**0.85**	0.63

Table 2: OOD Detection Performance (Average across datasets)

Scorer	AUROC ↑	FPR95 ↓
MSP	0.795 ± 0.008	0.556 ± 0.029
MaxLogit	0.805 ± 0.012	0.540 ± 0.035
Energy	$\mathbf{0.807 \pm 0.014}$	$\mathbf{0.538 \pm 0.036}$
Mahalanobis	0.779 ± 0.012	0.624 ± 0.033
ViM	0.773 ± 0.043	0.574 ± 0.058
NECO	$\mathbf{0.807 \pm 0.085}$	0.633 ± 0.149

As we can see, the NECO method scored really well, as it's expected due to its fitness to the end of training structure. The overall (across datasets) best performance was achieved by the energy method, in close proximity to the NECO scorer in terms of AUCROC. This result is interesting, as it reflects the soundness in the approach taken in the exam's paper [2] (by mixing up the two).

2.3 Notebook 3 : Neural Collapse Analyses (NC1 to NC4) & Notebook 4 : Neural Collapse Analysis NC5

(Located under *\notebooks\03_neural_collapse.ipynb* and *\notebooks\04_neural_collapse_ood.ipynb*)

Q3 & Q4

Study the Neural Collapse phenomenon at the end of training NC 1 to NC5

In this section, we take all the usual neural collapse properties and evaluate them during the saved epochs over TPT :

2.3.1 Definitions

Let $h(x) \in \mathbb{R}^d$ denote the penultimate-layer feature representation and M_c the empirical mean feature of class c. We define the global mean

$$\bar{M} = \frac{1}{C} \sum_{c=1}^{C} M_c$$

and the centered class means

$$\tilde{M}_c = M_c - \bar{M}, \qquad \tilde{M} = [\tilde{M}_1, \ldots, \tilde{M}_C] \in \mathbb{R}^{d \times C}.$$

The within-class and between-class covariance matrices are

$$S_W = \frac{1}{N} \sum_{c=1}^{C} \sum_{x \in c} (h(x) - M_c)(h(x) - M_c)^\top$$

$$S_B = \frac{1}{C} \tilde{M} \tilde{M}^\top$$

Since $\operatorname{rank}(S_B) \leq C - 1$, it is singular. Therefore we use the Moore–Penrose pseudoinverse $S_B^\dagger$ computed from the SVD of $\tilde{M}$.

NC1 — Within-Class Variability Collapse

$$\text{NC1} = \frac{1}{C} \operatorname{Tr}(S_W S_B^\dagger)$$

This quantity measures how concentrated features are around their class centers. When NC1 decreases, intra-class variability vanishes compared to inter-class separation. In the collapse regime, each class becomes a single point in feature space.

NC2 — Convergence to a Simplex Equiangular Tight Frame

Two complementary geometric properties are evaluated.

Equinorm:

$$\mathrm{CoV}(\|M_c\|) \to 0$$

All class centers lie on a sphere of identical radius.

Equiangularity:

Instead of computing pairwise angles directly, we measure the deviation of the normalized Gram matrix from that of a simplex ETF. This verifies that the angles between any two class means converge to the same value

$$\cos\theta = -\frac{1}{C-1}.$$

Geometrically, the class means become maximally separated points on a sphere.

NC3 — Self-Duality

$$\mathrm{NC3} = \|\hat{W}^\top - \hat{\tilde{M}}\|_F^2$$

where both matrices are Frobenius-normalized.

This evaluates alignment between classifier weights and class means. At convergence the linear classifier becomes equivalent to a nearest-prototype classifier in feature space.

NC4 — Nearest Class Center Rule

$$\mathrm{NC4} = 1 - \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\Big[\arg\max_c f_c(x_i) = \arg\min_c \|h(x_i) - M_c\|\Big]$$

This measures the disagreement between the neural network prediction and the nearest centroid classifier. Neural collapse predicts this value converges to zero: the network behaves as a geometric classifier.

In practice, we compute operational metrics approximating the theoretical limits, since exact equalities hold only asymptotically (infinite training time and width).

Note that we compute operational metrics that approximate the theoretical Neural Collapse definitions, since exact equalities hold only in the limit of infinite training time and infinite width.

2.3.2 Experiments

Table 3: Neural Collapse Metrics (Final Epoch)

Metric	Value
NC1 $\left(\frac{1}{C}\mathrm{Tr}(S_W S_B^\dagger)\right)$	5.50
NC2a (Equinorm: CoV of $\|M_c\|$)	0.033
NC2b (Equiangularity: ETF Gram deviation)	0.058
NC3 $\left(\|\hat{W}^\top - \hat{\tilde{M}}\|_F^2\right)$	0.037
NC4 (Network vs NCC disagreement)	≈ 0.000

2.3.3 Analysis of Neural Collapse Metrics

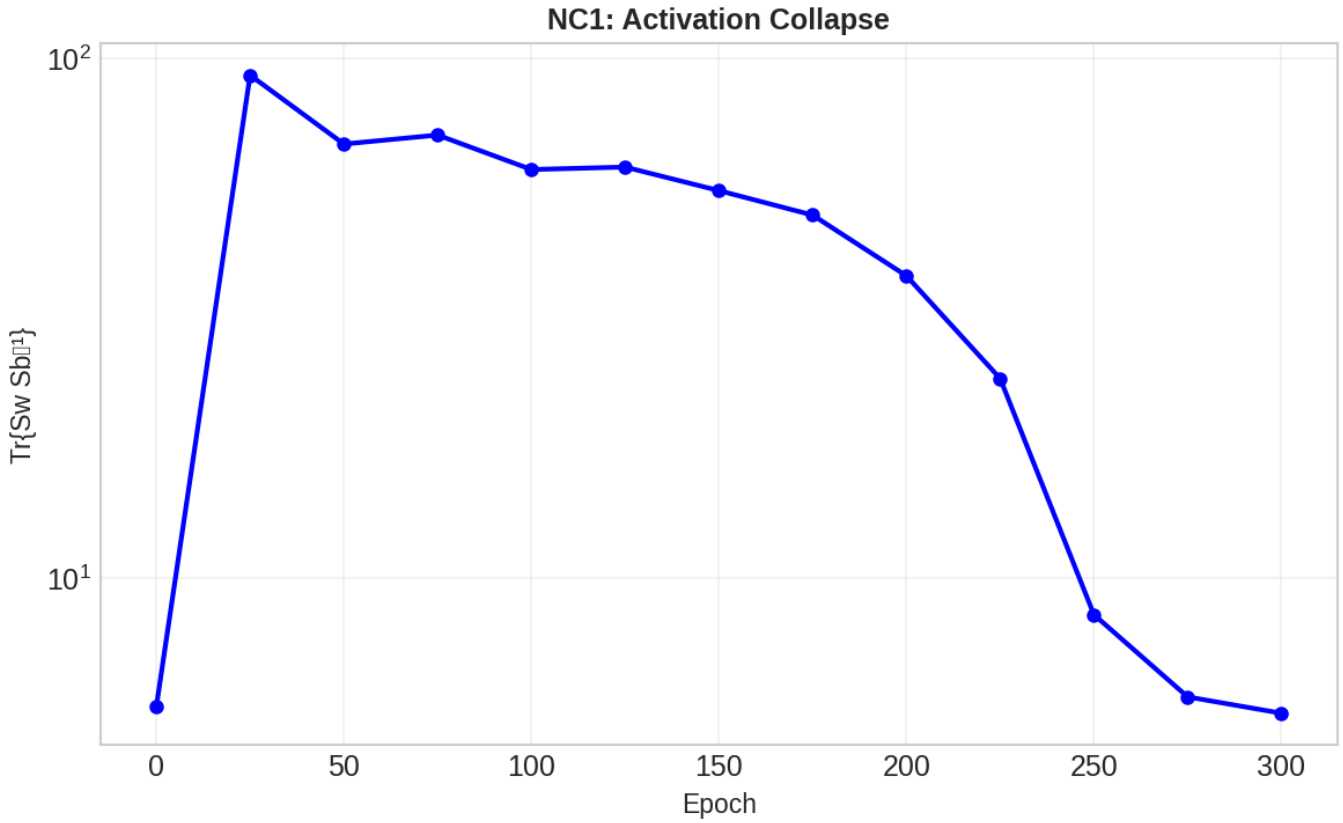


Figure 9: NC1

NC1 — Within-Class Variability Collapse The NC1 curve exhibits two clearly distinct regimes. During the early and intermediate stages of training the quantity $\frac{1}{C}\mathrm{Tr}(S_W S_B^{\dagger})$ first increases and then decreases slowly. This corresponds to the representation learning phase, where the network mainly focuses on separating decision boundaries rather than compressing classes.

After the terminal phase of training (when classification error is essentially zero), the metric drops sharply by nearly one order of magnitude. At this stage, optimization no longer improves accuracy but reorganizes the geometry of the representation.

This late rapid decrease indicates true within-class collapse: samples of the same class concentrate tightly around their class mean, turning each class into a single prototype in feature space. The network therefore transitions from learning decision boundaries to learning a symmetric representation structure.

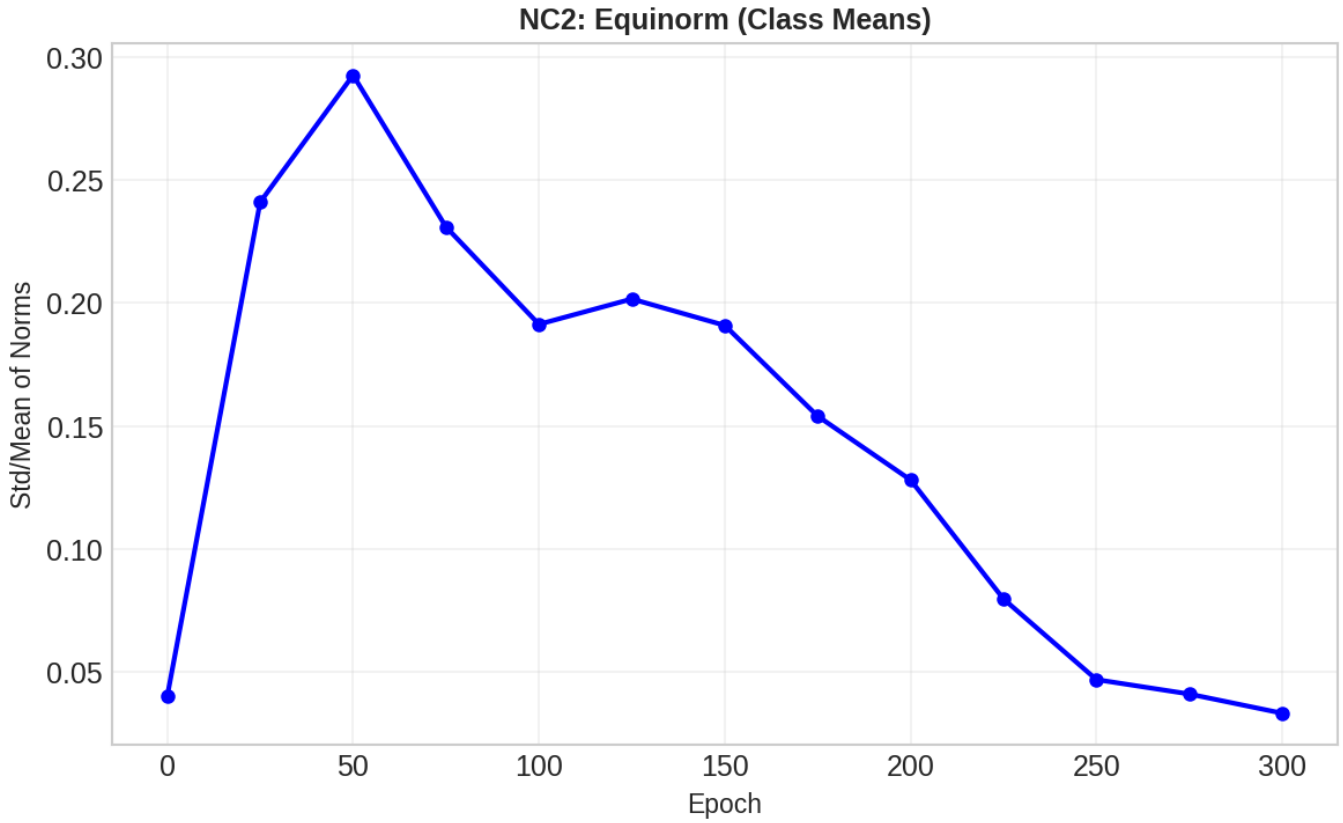


Figure 10: NC2a - Equinorm of M

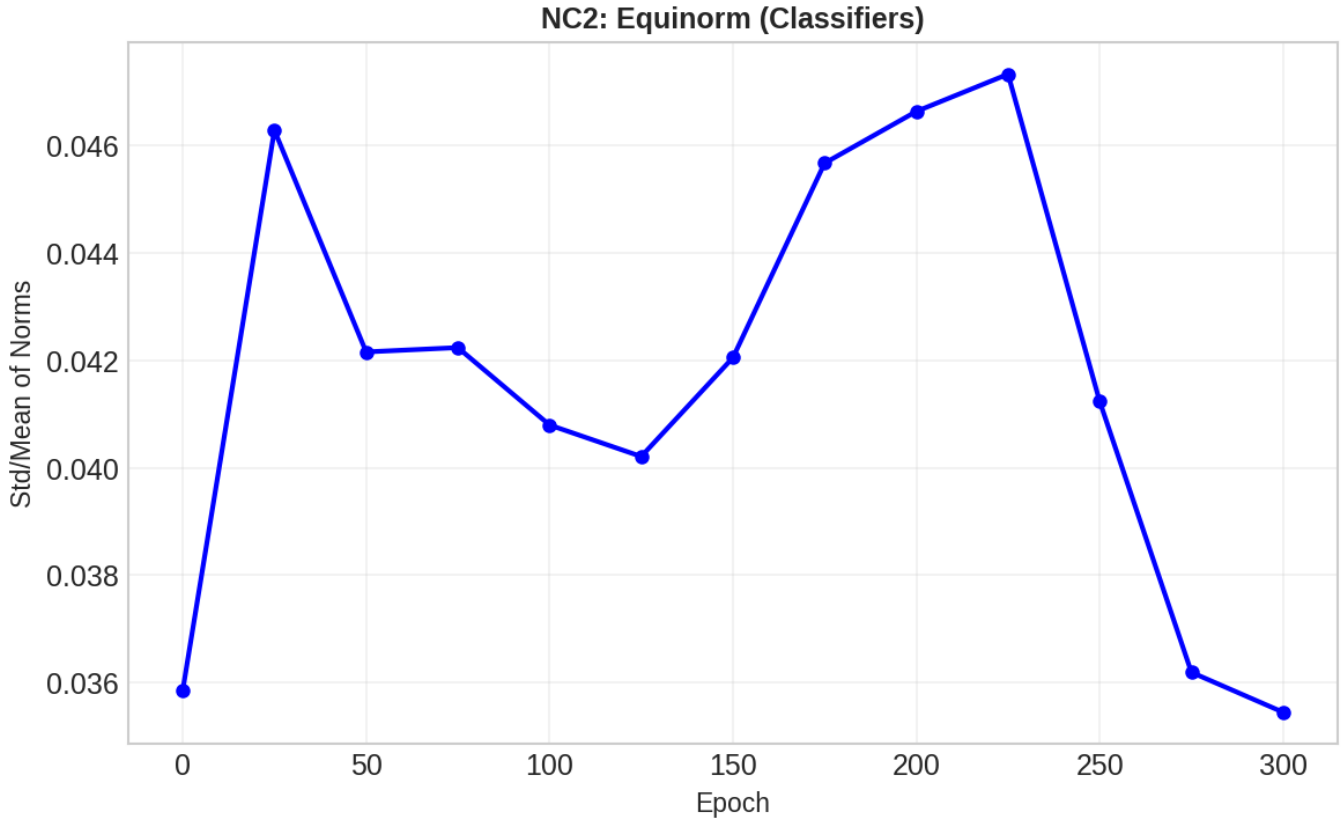


Figure 11: NC2a - Equinorm of W

NC2a — Equinorm (Class Means and Classifier Weights) We track the coefficient of variation of both the class mean norms and the classifier weight norms. During training, the two quantities progressively decrease and converge toward the same low-variance regime.

The classifier weights become nearly equinorm early in training, while the class means only reach this property during the terminal phase. After this point, both curves stabilize at similarly small values, indicating that the representation geometry and the classifier geometry become consistent.

This behavior shows that the feature space reorganizes so that all classes lie on a common hypersphere and the classifier weights match this spherical structure. In other words, the network not only separates classes but reshapes the representation so that the learned features and the linear classifier share the same symmetric geometry.

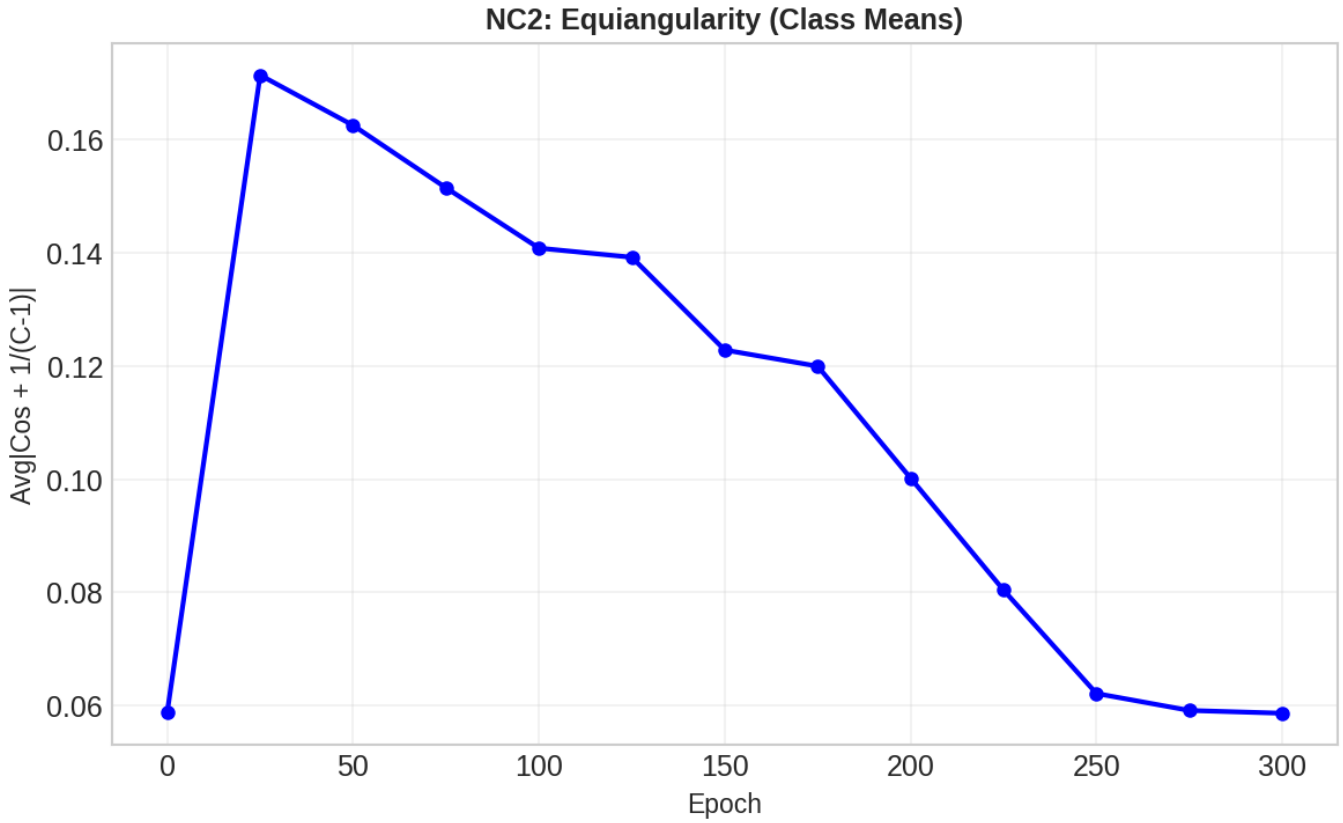


Figure 12: NC2a - Equiangularity of M

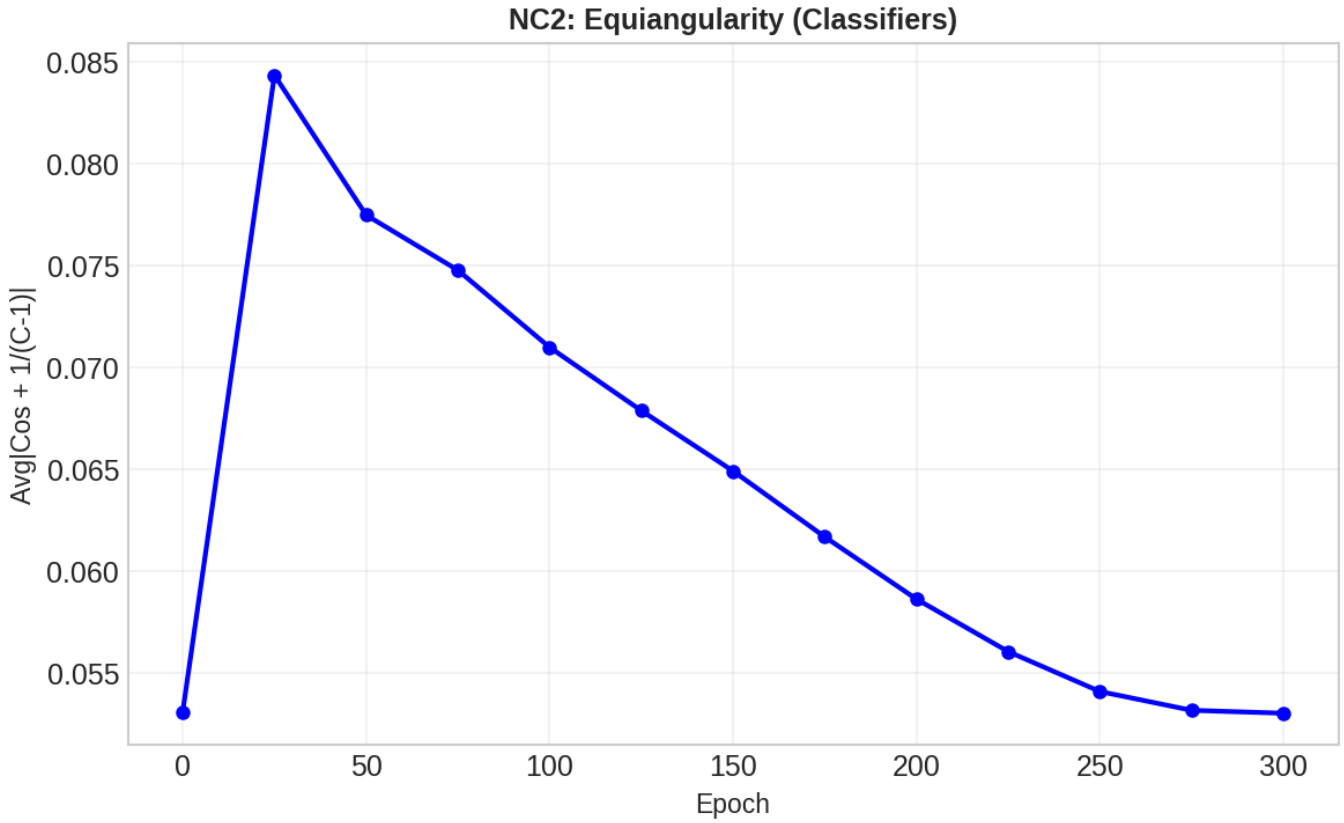


Figure 13: NC2a - Equiangularity of W

NC2b — Equiangularity (Simplex ETF Geometry) of Class Means and Classifier Weights We measure the deviation of the normalized Gram matrix from that of a simplex Equiangular Tight Frame. Both the class means and the classifier weights progressively decrease toward a common constant value.

The classifier weights approach equiangularity earlier, while the class means follow during the terminal phase of training. Eventually, both converge to the same regime, indicating that the representation adopts a highly symmetric configuration.

This corresponds to the formation of a simplex ETF: all class centers become equally separated directions in feature space, with pairwise angles approaching

$$\cos\theta = -\frac{1}{C-1}.$$

Geometrically, the network is no longer only minimizing classification error. Instead, it organizes the representation as a maximal separation packing problem on the hypersphere, distributing classes uniformly in all directions.

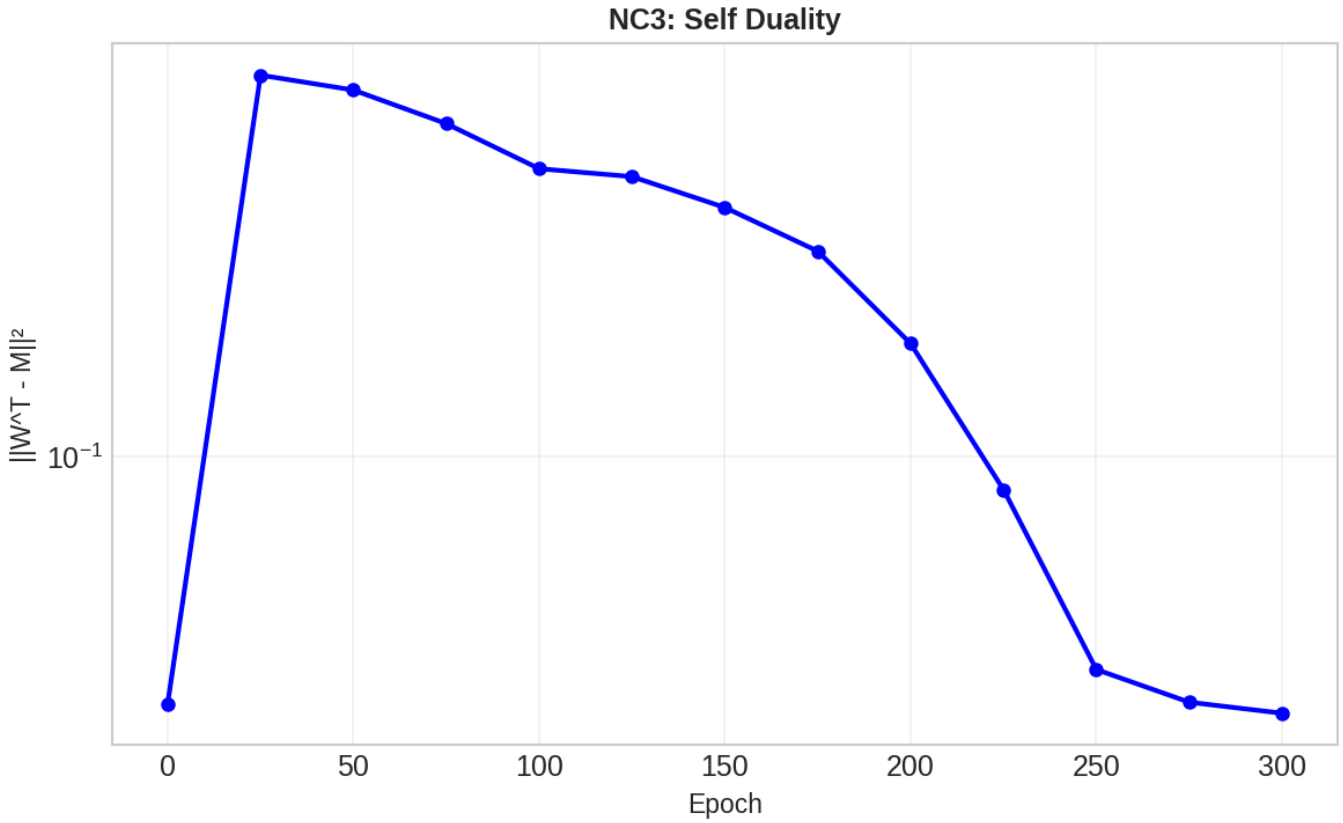


Figure 14: NC3 - Self-Duality

NC3 — Self-Duality The distance $\|\hat{W}^\top - \hat{\tilde{M}}\|_F^2$ decreases slowly during most of training and then drops sharply after the terminal phase. This indicates a qualitative change in the role of the classifier.

Before reaching zero training error, the last linear layer learns a conventional decision boundary separating feature regions. Once the network perfectly fits the data, optimization instead aligns classifier weights with class means.

As a result, each weight vector becomes parallel to its corresponding class prototype. The linear classifier no longer implements an arbitrary separating hyperplane but instead computes similarity to class centers. The decision rule therefore transitions from boundary-based classification to prototype matching.

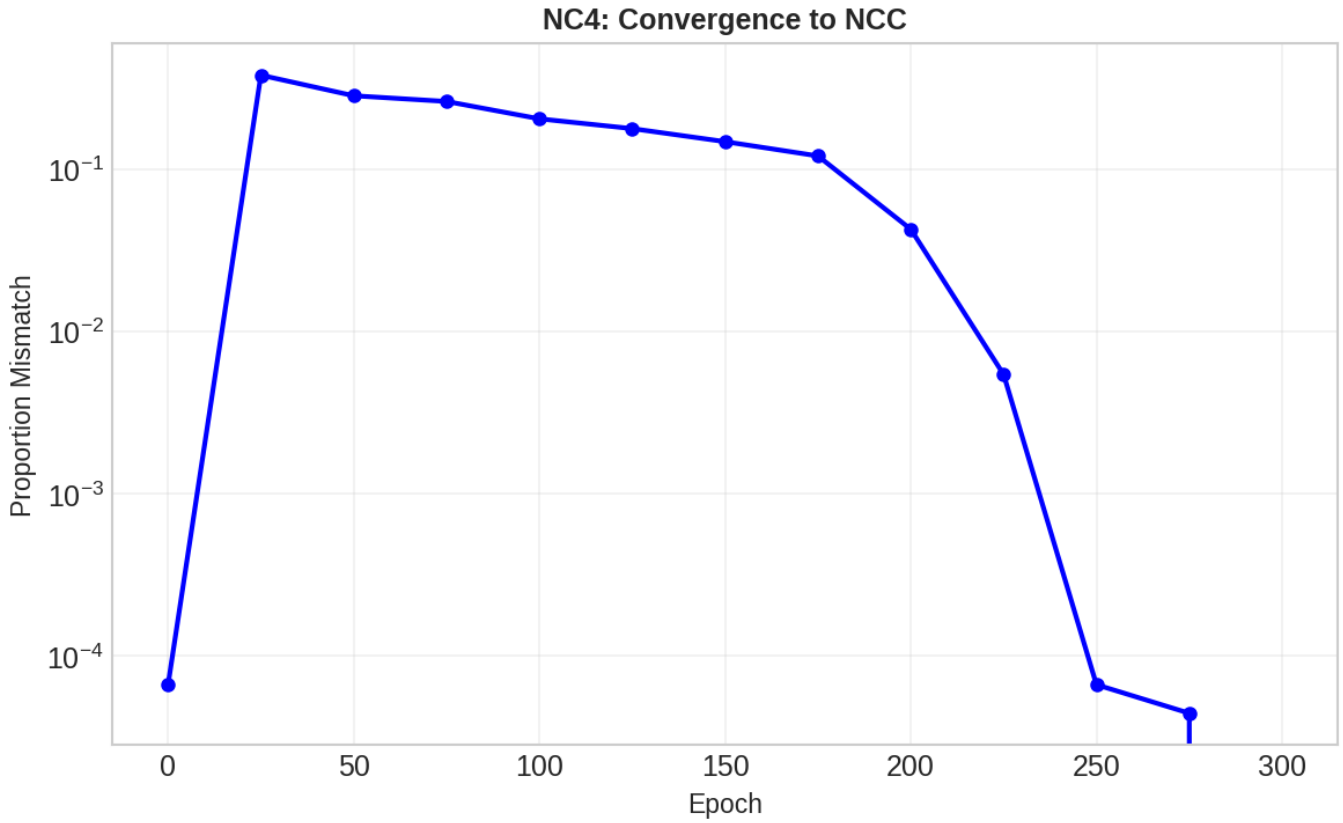


Figure 15: NC4 - NCC-Convergence

NC4 — Nearest Class Center Rule The disagreement between the neural network predictions and the nearest centroid classifier decreases dramatically after the terminal phase of training, dropping by several orders of magnitude.

Importantly, this reduction occurs after classification accuracy has already saturated. Therefore the network is no longer learning to classify better, but rather simplifying its decision rule.

At convergence, the neural network behaves identically to a nearest class center classifier in the learned feature space. The deep model thus reduces to a feature extractor followed by Euclidean distance comparison to class prototypes.

This confirms the final stage of Neural Collapse: once representations become symmetric and weights align with class means, the optimal decision rule is purely geometric.

2.3.4 NC5 — ID/OOD Orthogonality

Neural Collapse predicts that, at convergence, in-distribution (ID) features concentrate in a low-dimensional class subspace, while out-of-distribution (OOD) samples tend to occupy directions that are (approximately) orthogonal to this ID subspace. We evaluate NC5 by measuring an orthogonality deviation score between OOD features and the ID class-mean subspace during training.

NC5 Overview: Joint Summary with OOD

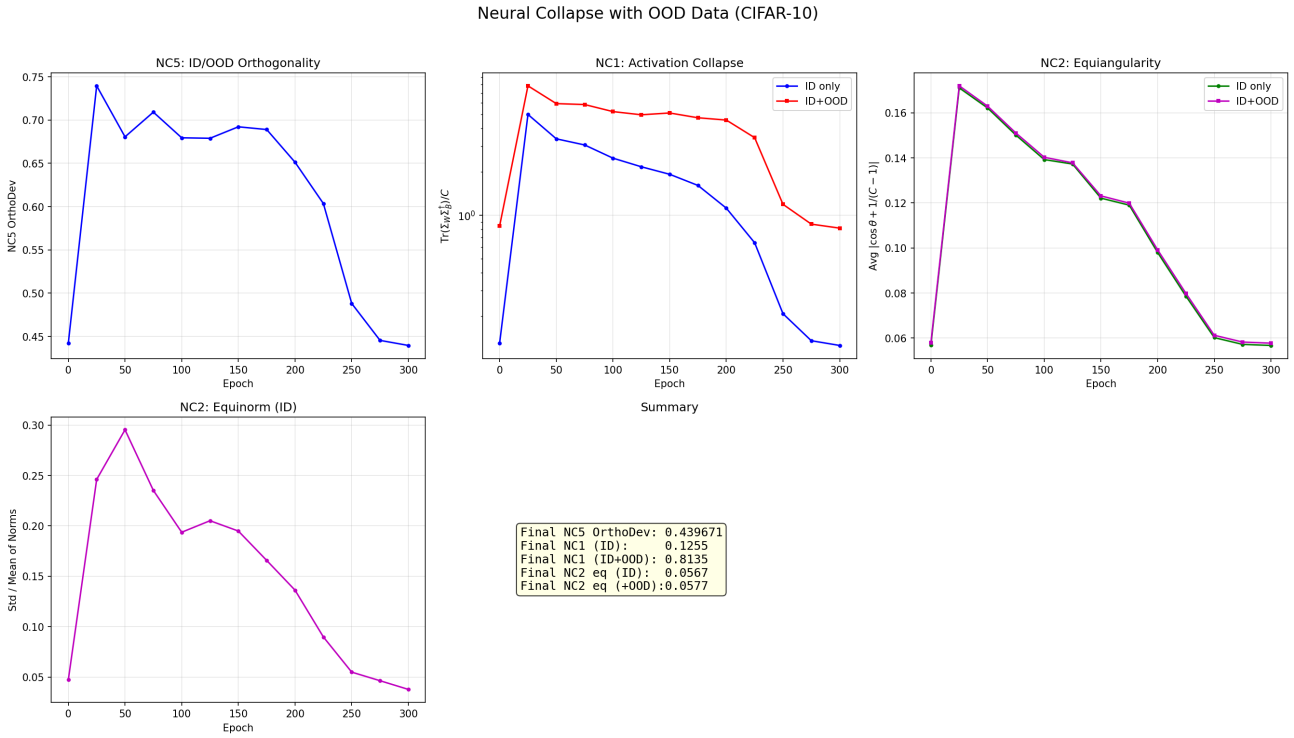


Figure 16: Neural Collapse metrics measured with OOD data. The panels summarize NC5 orthogonality and the influence of adding OOD samples on NC1 and NC2.

This overview figure highlights two important facts. First, NC2 metrics remain nearly unchanged when OOD samples are added, indicating that the simplex ETF geometry is a property of the class means and classifier weights, not of the full input distribution. Second, NC1 increases under ID+OOD evaluation because OOD features do not collapse around any class mean, thus inflating the apparent within-class variability when considering the union of ID and OOD samples.

Global Orthogonality Dynamics Across OOD Datasets

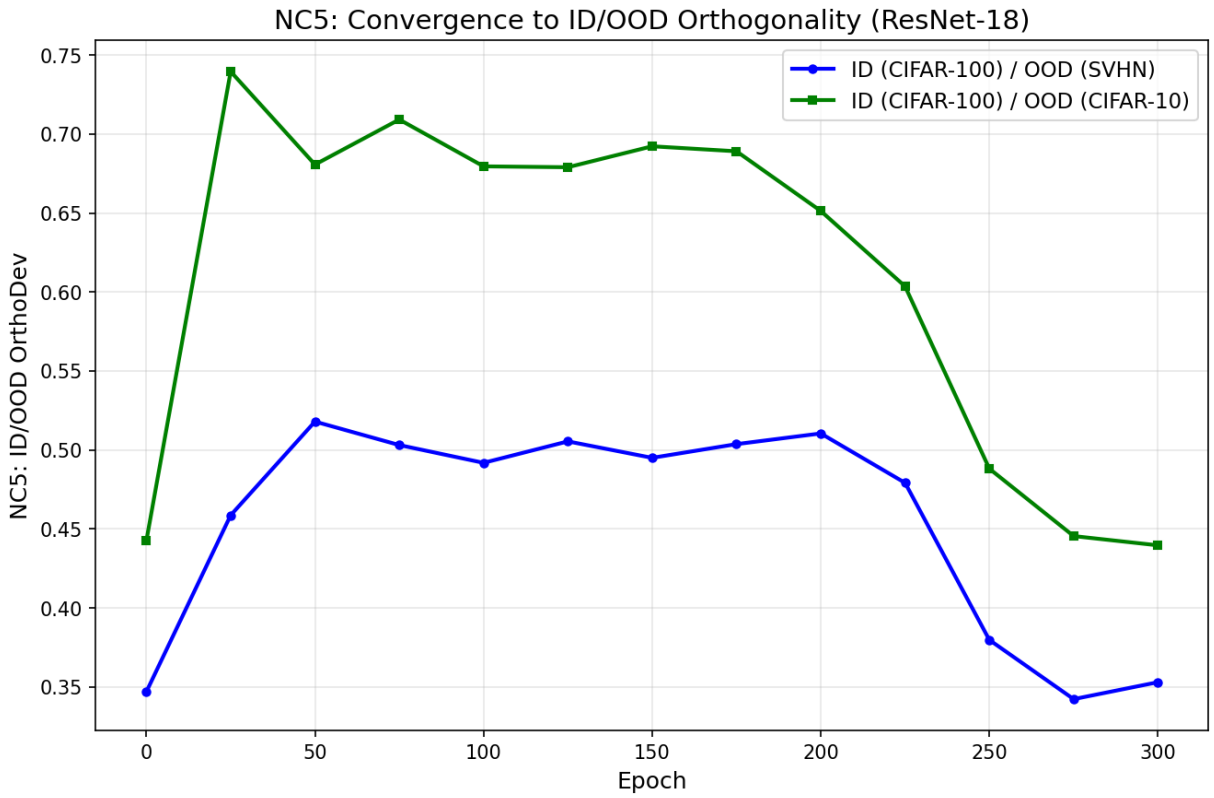


Figure 17: Evolution of the NC5 orthogonality deviation for different OOD datasets.

The orthogonality deviation decreases substantially only after the terminal phase of training. This suggests that the network first learns to classify (ID decision boundaries) and only later reshapes the representation geometry. In the late regime, the learned feature space behaves as if it were decomposed into an ID class subspace and complementary directions where OOD samples are pushed.

Temporal Convergence of NC5 (Aggregate)

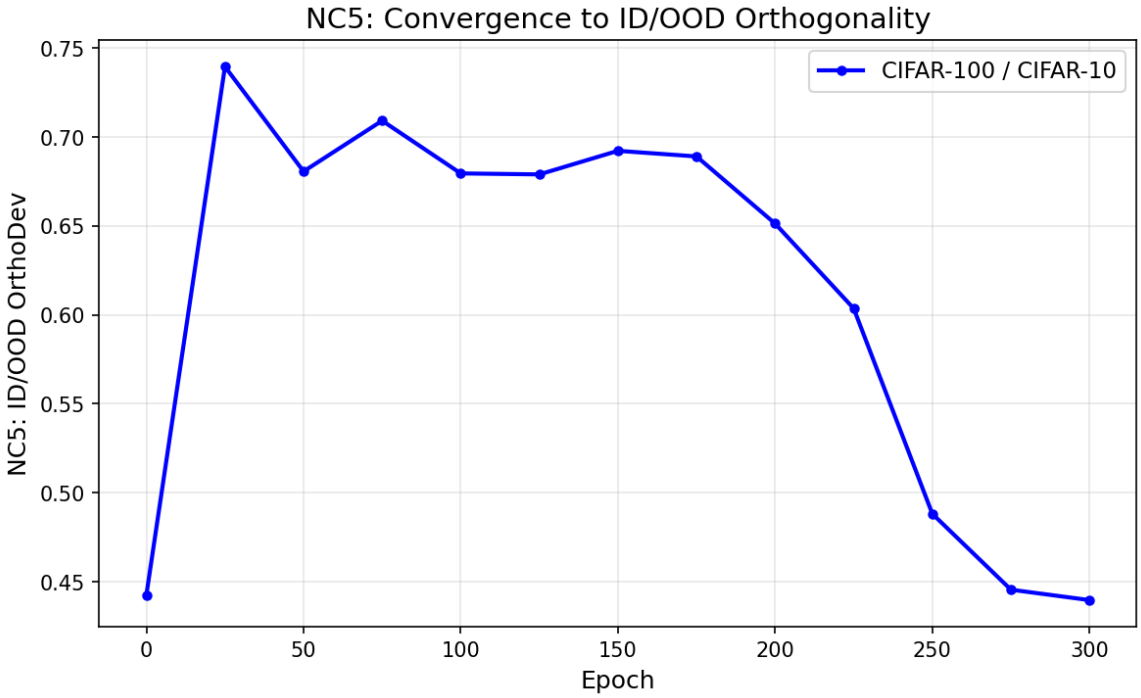
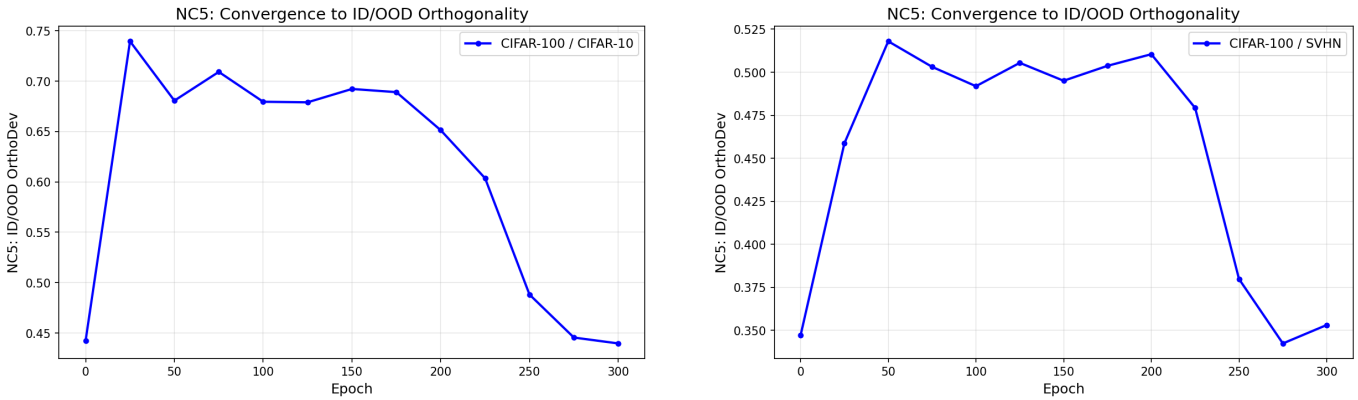


Figure 18: Aggregate NC5 convergence: orthogonality deviation during training.

The timing of the decrease matches the ordering predicted by Neural Collapse: NC1–NC4 mature first (class clusters collapse and become symmetric), then NC5 emerges as a secondary effect where the model separates the ID manifold from the ambient feature space. In other words, the representation becomes both *class-structured* and *support-aware*.

Dataset-Specific NC5 Convergence: CIFAR-10 vs SVHN



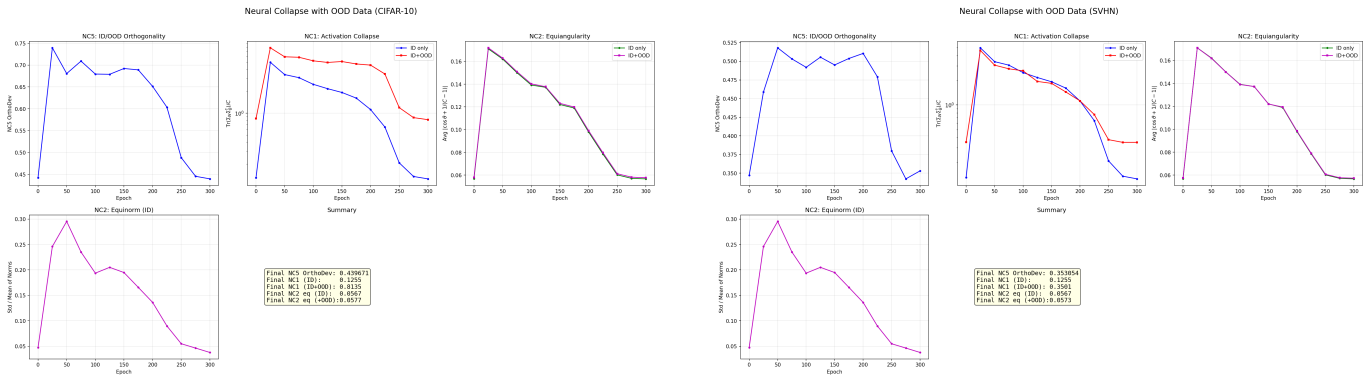
(a) CIFAR-10 as OOD

(b) SVHN as OOD

Figure 19: NC5 convergence per OOD dataset.

The orthogonality effect is stronger for SVHN than for CIFAR-10. This is consistent with semantic distance: SVHN (digits) is substantially different from CIFAR-100 (natural images), whereas CIFAR-10 shares similar low-level statistics and partially overlaps the ID feature subspace. Consequently, CIFAR-10 remains less separable in the learned geometry, while SVHN is pushed farther away.

Effect of OOD Samples on NC1 and NC2 (Detailed Comparison)



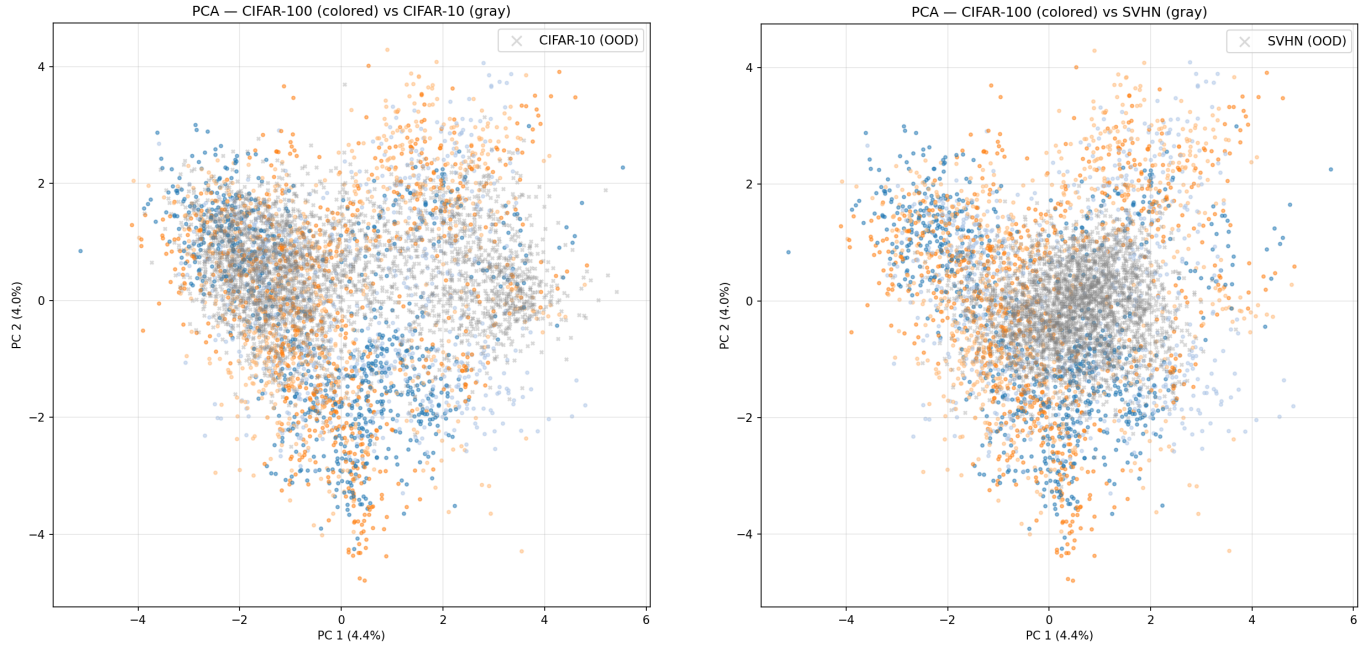
(a) CIFAR-10 as OOD (b) SVHN as OOD

Figure 20: Influence of adding OOD samples on NC1 and NC2.

Across both OOD datasets, NC2 (equiangularity) curves are nearly identical for ID-only and ID+OOD evaluation, supporting the claim that the simplex ETF structure is an intrinsic organization of class prototypes. In contrast, NC1 is consistently higher with ID+OOD because OOD features cannot be assigned to a collapsed class cluster and therefore increase variance measured around class means.

PCA Visualization of ID vs OOD Feature Spaces

To complement the quantitative NC5 analysis, we visualize the penultimate-layer features using PCA (2 components), projecting both ID (CIFAR-100, colored by class) and OOD samples (gray crosses) onto the same coordinate system. The PCA is fitted on CIFAR-100 features only, so the axes represent the directions of maximal variance in the ID feature space.



(a) CIFAR-10 as OOD (b) SVHN as OOD

Figure 21: PCA projection of penultimate-layer features. CIFAR-100 (ID) samples are shown as colored dots; OOD samples are shown as gray crosses.

The two plots reveal a striking contrast. In Figure 21b, SVHN features (gray) concentrate in a tight, compact cluster near the center of the ID distribution, occupying a much smaller region than the CIFAR-100 features. The ID samples spread outward into distinct class clusters along the periphery, while SVHN samples barely overlap with them. This is a direct visual manifestation of NC5: the network has learned

to map semantically distant OOD inputs into directions that carry little variance in the ID class subspace. Since the PCA axes are defined by ID variance, SVHN features projecting near the origin means they lie approximately orthogonal to the ID class-mean subspace in the full 512-dimensional feature space.

In Figure 21a, CIFAR-10 features (gray) are far more dispersed and overlap substantially with the ID distribution. Many CIFAR-10 samples fall among or near CIFAR-100 class clusters, making them much harder to separate geometrically. This is consistent with the weaker NC5 orthogonality deviation observed for CIFAR-10 compared to SVHN (Figure 17), and explains why all OOD scoring methods, including NECO, achieve lower AUROC on CIFAR-10 than on SVHN (Table 1).

The low explained variance of the first two principal components (4.4% and 4.0%) is itself a consequence of Neural Collapse: with $C = 100$ classes arranged in a near-simplex ETF, the representation distributes variance roughly uniformly across $C-1 = 99$ dimensions rather than concentrating it along a few axes. This is the expected signature of a collapsed representation where class means are maximally and symmetrically separated.

Geometric Interpretation and Link to OOD Scoring

Neural Collapse therefore has two simultaneous geometric outcomes. First, it organizes ID classes into a maximally symmetric configuration (NC1–NC4). Second, it implicitly learns the support of the training distribution by separating OOD samples into complementary directions (NC5). This gives a geometric explanation for collapse-based OOD detectors such as NECO: they detect samples that deviate from the collapsed ID class subspace rather than relying only on confidence or logits.

2.4 Notebook 5 : Theoretical Interpretation

(Located under *\notebooks\05_neural_collapse_earlier_layers.ipynb*)

Q6

Neural Collapse on Earlier Layers

We now investigate how Neural Collapse properties emerge across network depth. For each checkpoint, we extract features after Global Average Pooling at five points in the ResNet-18 architecture (with C = 100, since it's the number of classes):

Layer	Feature dim D	Spatial	D vs $C = 100$
layer1	64	32×32	$D < C$
layer2	128	16×16	$D > C$
layer3	256	8×8	$D > C$
layer4	512	4×4	$D > C$
penultimate	512	(GAP)	$D > C$

NC5 is computed using SVHN as the OOD dataset. NC3 (self-duality) is only defined at the penultimate layer, since it requires the classifier weight matrix W.

2.4.1 Temporal Dynamics: NC1–NC5 Across Layers and Epochs

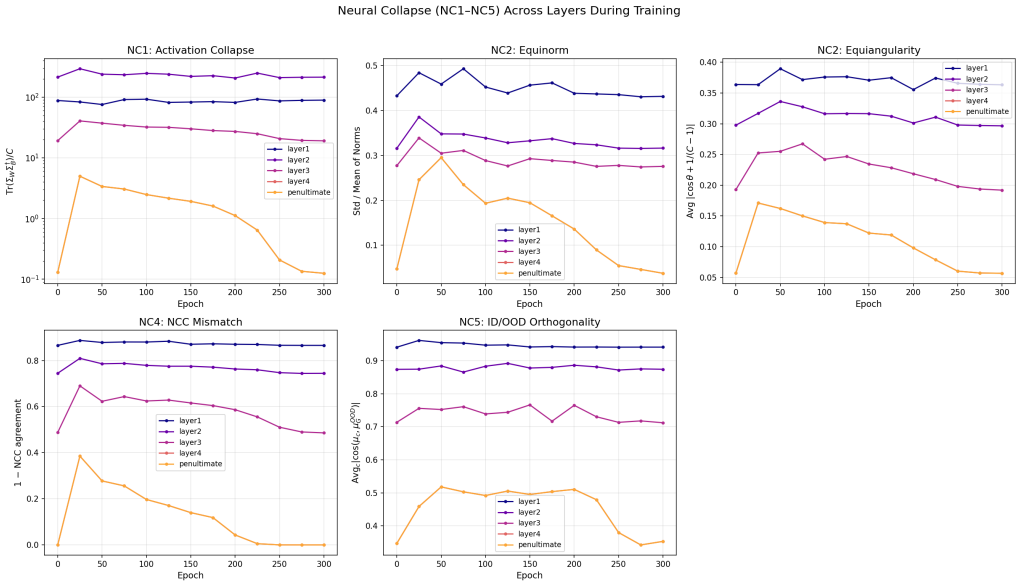


Figure 22: NC1–NC5 across epochs for each layer. Each curve corresponds to one layer; the penultimate layer is shown dashed.

The line plots reveal several key patterns. For NC1, all layers exhibit a rise-then-fall trajectory, but the descent is steeper and begins earlier for deeper layers: the penultimate layer drops by nearly two orders of magnitude between epoch 150 and 300, while layer1 barely decreases. For NC2 (both equinorm and equiangularity), the curves fan out with depth. Deeper layers start at comparable values but converge to much lower ones by the terminal phase of training (TPT). NC4 shows a clear layerwise ordering that becomes apparent only after epoch 200: the penultimate layer drops to near zero while earlier layers plateau at high values. NC5 follows a similar depth ordering, with the penultimate and layer4 curves descending most steeply during TPT.

The overall picture confirms that Neural Collapse propagates from the last layer backward: the penultimate layer collapses first, followed by layer4, then layer3, while layers 1 and 2 exhibit only partial collapse.

2.4.2 Layerwise Heatmaps

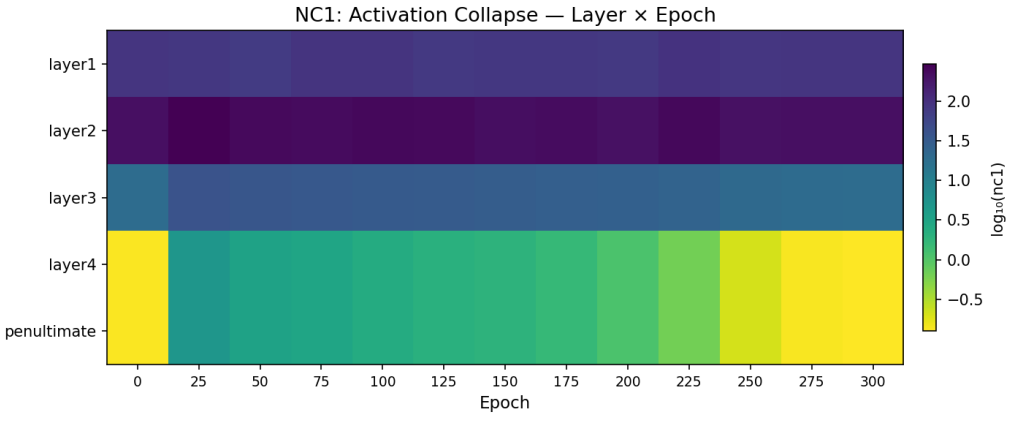


Figure 23: NC1 heatmap (layer × epoch). Color encodes $\log_{10}(\text{NC1})$; cooler tones indicate stronger collapse.

NC1 — Within-Class Variability Collapse Early layers (layer1, layer2) maintain high NC1 values ($\log_{10}(\text{NC1}) \approx 1.5$–$2.0$) throughout training, indicating that features at these depths never fully concentrate around their class means. This is expected for layer1 in particular, where $D = 64 < C = 100$ makes the between-class covariance matrix severely rank-deficient and NC1 ill-conditioned.

Deeper layers (layer4, penultimate) show a clear transition during TPT: NC1 drops sharply after epoch 225, visible as a color shift from blue to yellow tones in the bottom rows. Notably, NC1 does not decrease

monotonically with depth at all epochs, during intermediate training, layer2 and layer3 can exhibit values comparable to or higher than layer1, suggesting temporary intra-class expansion before the final compression in later stages.

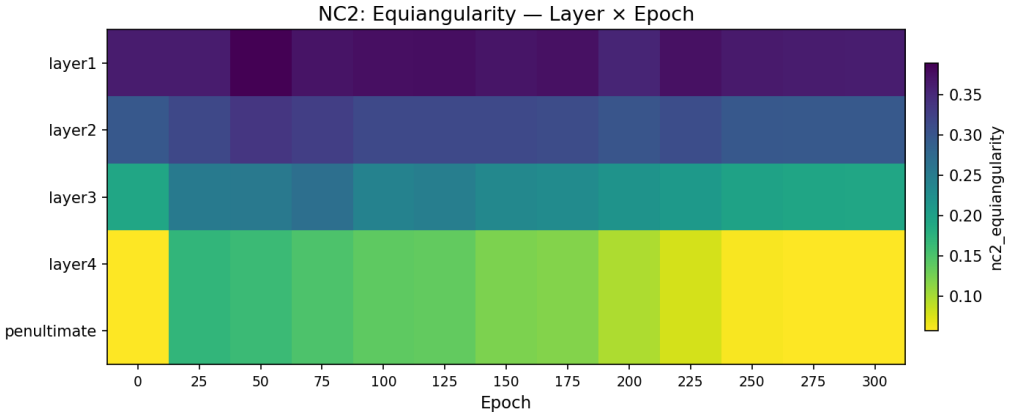


Figure 24: NC2 equiangularity heatmap (layer × epoch).

NC2 — Equiangularity The equiangularity heatmap shows a clear depth gradient. Early layers display persistently high deviation from the simplex ETF structure (≈ 0.25–0.35) across all epochs and never approach the ideal equiangular configuration, consistent with low-dimensional feature spaces being unable to support a 100-vertex simplex ETF.

Deeper layers progressively converge toward lower deviation. The penultimate layer reaches the lowest values (≈ 0.06) by epoch 300, with the improvement accelerating during TPT. Layer4 follows closely, while layer3 shows intermediate behavior. This confirms that the simplex ETF geometry is primarily a property of deeper representations and that each successive residual stage contributes to aligning class means toward this maximally symmetric configuration.

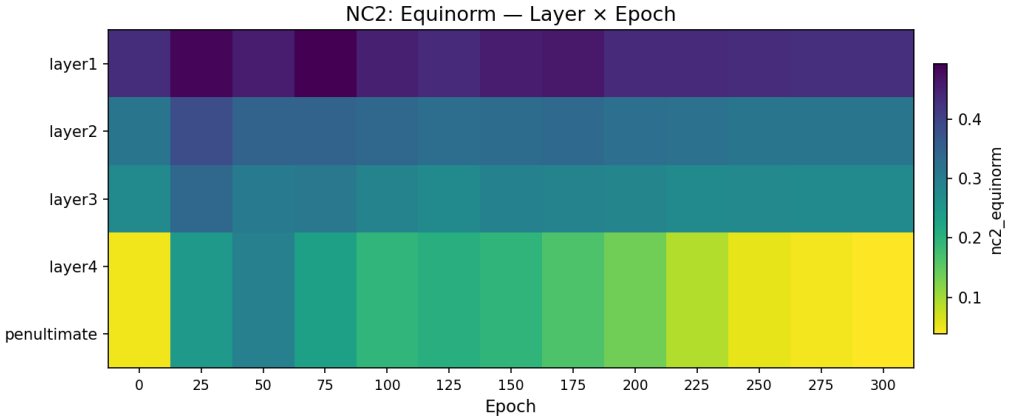


Figure 25: NC2 equinorm heatmap (layer × epoch).

NC2 — Equinorm The equinorm heatmap is qualitatively similar to the equiangularity one but with a sharper contrast between the earliest layers and the rest. Layer1 shows an extremely high coefficient of variation of class mean norms (≈ 0.4–0.5), meaning class centers are far from lying on a common hypersphere-.This is also attributable to the dimensional bottleneck ($D < C$).

From layer2 onward, equinorm values decrease substantially, and the penultimate layer achieves the lowest values by the end of training. An important observation is that layer1 shows almost no improvement across epochs: unlike deeper layers, the equinorm property at this depth is structurally constrained rather than being a matter of optimization. This supports the interpretation that Neural Collapse requires sufficient representational capacity ($D \geq C$) to manifest fully.

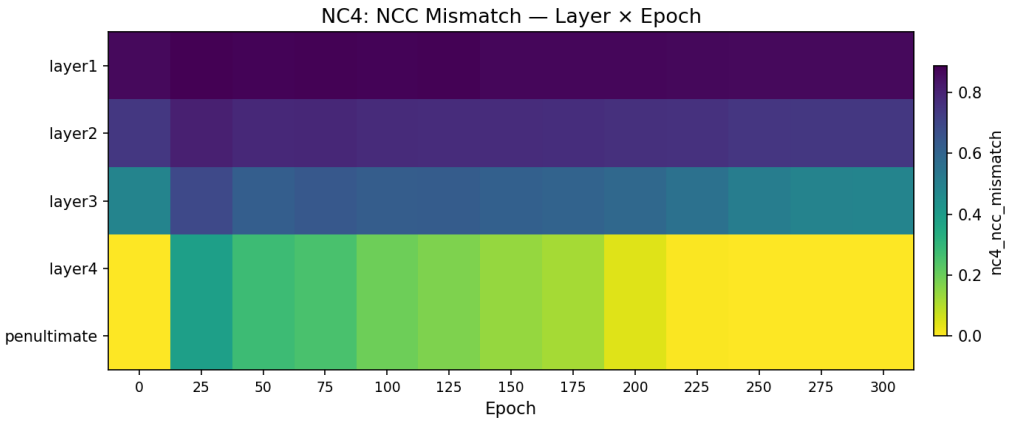


Figure 26: NC4 NCC mismatch heatmap (layer × epoch).

NC4 — NCC Mismatch NC4 shows the most dramatic depth dependence among all metrics. Layer1 and layer2 maintain high NCC mismatch (0.6–0.8) throughout the entire training process, meaning a nearest-class-center classifier built on early-layer features would disagree with the network's predictions for the majority of samples.

Layer3 shows intermediate behavior, with mismatch decreasing to approximately 0.3–0.4 by epoch 300. Layer4 and the penultimate layer converge toward near-zero mismatch during TPT, confirming that the equivalence between the neural network and a nearest-centroid classifier is concentrated at the final representation stages. The sharp transition visible in the bottom rows after epoch 225 mirrors the penultimate-only NC4 analysis (Figure 15).

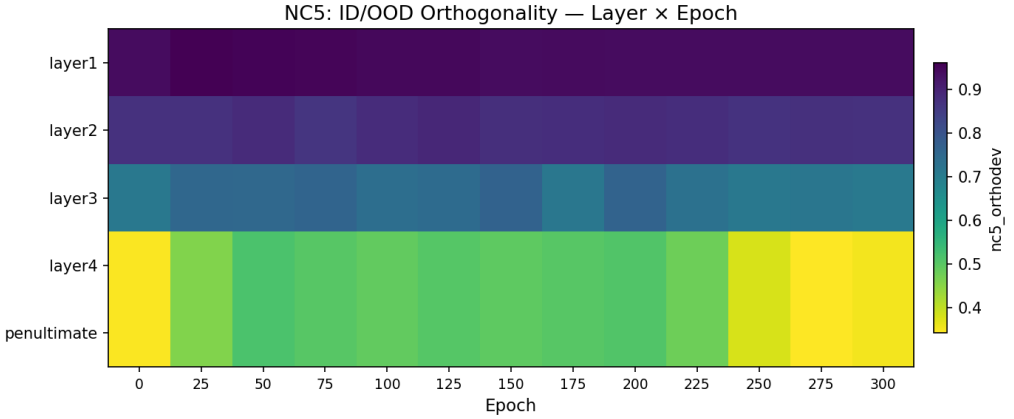


Figure 27: NC5 orthogonality deviation heatmap (layer × epoch). OOD dataset: SVHN.

NC5 — ID/OOD Orthogonality The NC5 heatmap reveals that ID/OOD orthogonality also improves with depth, although the gradient is less steep than for NC4. Early layers already exhibit moderate orthogonality deviation (≈ 0.7–0.9), suggesting that even shallow features partially distinguish ID from OOD inputs based on low-level statistics. However, the explicit geometric separation into complementary subspaces is refined by deeper layers.

The penultimate layer achieves the strongest orthogonality (lowest deviation, ≈ 0.3–0.4) by the end of training. The temporal dynamics of NC5 across layers mirror those of the other metrics: improvement is concentrated in TPT and is more pronounced for deeper layers, consistent with NC5 emerging as a secondary geometric consequence of NC1–NC4 maturation.

2.4.3 Summary Across Network Depth (Epoch 300)

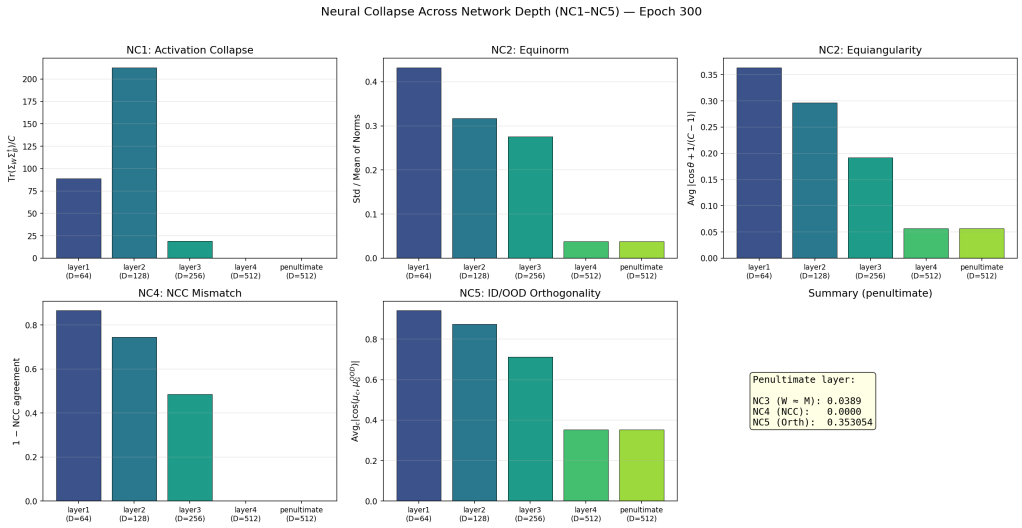


Figure 28: NC1–NC5 at each layer for the final checkpoint (epoch 300). The summary panel reports penultimate-layer values for NC3, NC4, and NC5.

The bar chart at epoch 300 provides a final snapshot of layerwise NC metrics. NC1 decreases approximately monotonically from layer1 to the penultimate layer (with the caveat that layer1's value is inflated by dimensional ill-conditioning). NC2 equinorm and equiangularity both decrease monotonically with depth, confirming that the simplex ETF geometry becomes progressively more structured through the network. NC4 shows the starkest contrast: early layers retain high NCC mismatch (> 0.6) while the penultimate layer reaches essentially zero. NC5 follows a similar monotonic improvement with depth.

These results collectively demonstrate that Neural Collapse is not an all-or-nothing phenomenon localized at the last layer. Rather, it emerges *progressively* through network depth, with each residual stage contributing incremental geometric structure toward the optimal simplex ETF codebook. The penultimate layer collapses first and most strongly, layer4 follows closely, while layers 1–2 remain only partially collapsed due to the fundamental constraint $D < C$ or $D \approx C$.

Q7

Mechanistic Interpretability

2.4.4 Initial Feature State and Functional Initialization

(Notions of theoretical interpretability were mainly taken from [3] and [4]).

We begin by considering the initial feature extractor, which maps the input image $x \in \mathbb{R}^{H\times W\times C}$ to an initial representation

$$x_0 = \phi_{\text{init}}(x) \tag{1}$$

through convolutional layers without residual connections.

This initial stage defines the first shared representational state upon which all subsequent residual transformations operate. The architecture was designed (as per explained in the Model Architecture subsection) such that the receptive field prior to the classifier head remains comparable to the input image size, ensuring that x_0 captures global semantic structure while preserving sufficient spatial resolution for progressive abstraction.

From a mechanistic perspective, this stage establishes the initial point in the model's functional space. Subsequent residual blocks can then be interpreted as iterative functional refinements of this shared state, rather than independent representational remappings.

2.4.5 Residual Blocks as Iterative State Refinement Operators

Each residual block implements a transformation of the form

$$x_{k+1} = x_k + f_k(x_k), \tag{2}$$

where f_k represents the residual function implemented by the convolutional layers within the block.

By recursion, after n residual blocks, the representation becomes:

$$x_n = x_0 + \sum_{i=0}^{n-1} f_i(x_i). \tag{3}$$

Under the approximation that each f_i operates on a slowly drifting representation (a standard assumption in residual network analysis [5]), this can be interpreted as

$$x_n \approx x_0 + \sum_{i=0}^{n-1} f_i(x_0). \tag{4}$$

This additive structure is central to mechanistic interpretability. It implies that the representation evolves through incremental, composable updates rather than complete overwriting. Each residual block contributes an additive vector that refines the representation while preserving previously computed information.

This structure closely resembles additive functional expansion methods such as boosting, where the model progressively refines its representation through incremental corrections. In boosting, a predictor takes the form:

$$F_n(x) = F_0(x) + \sum_{i=1}^{n} h_i(x), \tag{5}$$

with each h_i correcting residual errors of the previous stage.

Similarly, in residual networks, each block refines the representation in directions that improve class separation and structural alignment.

2.4.6 Neural Collapse as Progressive Codebook Formation

Let $h_k(x)$ denote the pre-logit representation after residual block k. Neural Collapse theory predicts that, at convergence, the last layer's representation (k=4 for resnet18) should exhibit highly structured geometric properties. Due to the facts that our results 22 show that this somewhat occurs in the intermediate layers. If we look at the absolute value of each metric, they are more or less constant the earlier we go into each layer, as well as bigger in absolute value. We also see that the degree for which each layer's metrics decrease when closing in to TPT is more pronounced the deeper we go. As we can see in 22, layer 4 has a steeper descent the closer we get to TPT, while layer 3 is somewhat slower, but faster than layer 2, etc.

We claim here that, in accordance to what the Neural Collapse paper ([6]) provides as a interpretation to such phenomenon, the simplex ETF is likely to occur here as well, with a few nuances. First of all, the fact that the intermediate representation's dimension is bigger than the amount of classes thoroughly negates the possibility of perfect class separation per vertex, as shown in 28, this is actually the metric that drifted furthest from 0, which stands in stark contrast to what's seem in the NC2 metrics, that actually track the geometric structure of the simplex structure. We theorize that what's happening is that, throughout the training procedure, the closer we get to TPT, the more the entire structure collapses to intermediate ETF representations, with this happening first to the deeper layers. The line of reasoning behind this, in the context of the "iterative state refinement operators" interpretability framework, is that, at the end of each resnet block (which is what we measured) - composed of 2 residual blocks - the dimensionality reduction - alongside the doubling of convolutional features/channels and a increase in its receptive field - is guided by the mapping induced by the simplex ETF structure, making it a **optimal codebook for dimensionality reduction**.

The optimality here stems from a somewhat intuitive understanding of how optimal one can be when talking about mapping from a higher to a lower dimensional space. It's likely that something close to an orthogonal projection to such a structure is done here, similar to what happens in the moore-penrose pseudoinversion process. We'd have to further investigate though, in order to make this claim, likely in terms

of how the mean square error between subsequent projections work in this context of such "codebooks" (obviously across many more experiments with residual connections).

Mechanistically, to conclude, this suggests that residual blocks in TPT do not merely increase feature dimensionality but also iteratively improve the alignment between the representation and the optimal classification geometry.

3 References

References

[1] Mouïn Ben Ammar, Nacim Belkhir, Sebastian Popescu, Antoine Manzanera, and Gianni Franchi. Neco: Neural collapse based out-of-distribution detection, 2024.

[2] Restuccia-Group. Encore: Efficient neural collapse research framework. GitHub repository, 2024. https://github.com/Restuccia-Group/ENCORE.

[3] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. https://distill.pub/2020/circuits/zoom-in.

[4] Neel Nanda. A comprehensive mechanistic interpretability explainer & glossary. Online; accessed 2026-02-19, 2023. https://www.neelnanda.io/mechanistic-interpretability/glossary.

[5] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *CoRR*, abs/1806.07366, 2018.

[6] Vardan Papyan, X. Y. Han, and David L. Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *CoRR*, abs/2008.08186, 2020.