Practical Work: Out-of-Distribution Detection, OOD Scoring Methods, and Neural Collapse

Marie DIDIER, Damien LEGRAND

February 20, 2026

Abstract

This report details the training of a ResNet-18 classifier on CIFAR-100 and evaluates several post-hoc Out-of-Distribution (OOD) detection methods (MSP, Max Logit, Mahalanobis, Energy, and ViM). We also analyze the Neural Collapse (NC) phenomenon during the Terminal Phase of Training (TPT) and implement NECO, an OOD detection score inspired by this geometric behavior.

Contents

1 Introduction

1.1 Objective

The goal of this project is to study how a ResNet-18 classifier behaves at the end of training on CIFAR-100, and how its internal representations can be used for Out-of-Distribution (OOD) detection. We compare standard post-hoc OOD scoring methods and analyze the Neural Collapse phenomenon.

1.2 The Problem: Overconfidence

Deep neural networks often assign high probabilities to inputs it has never seen before and on the edge of the original distribution. This overconfidence is a major issue for deploying models in real-world scenarios, for example a system might not flag the need for manual override, even though it has never seen such data before. OOD detection addresses this by quantifying uncertainty, in order to flag or reject samples that don't belong to the training distribution.

1.3 Why Neural Collapse is Interesting

Neural Collapse shows that at the terminal phase of training, a network's latent representations reorganize into a highly structured, symmetrical geometry where samples of the same class perfectly cluster together. This predictable geometry can be highly useful for OOD detection:

- **A Strict Mathematical Baseline:** It provides a rigorous, low-dimensional framework (a Simplex) that describes exactly how known In-Distribution (ID) data should be represented.
- **Geometric Anomaly Detection:** Because the ID feature space is so organized, OOD samples naturally deviate from this structure, often falling orthogonal to the known class centers. We can exploit this with methods like NECO, which detect OOD data by measuring how poorly a test sample projects on this expected ID subspace.

2 Training ResNet-18 on CIFAR-100

2.1 Architecture modifications

The standard ResNet-18 was designed for ImageNet images (224×224). Applying it directly to CIFAR-100 (32×32) would reduce the feature map too much, so the model wouldn't learn the fine-grained local patterns necessary to distinguish between 100 different classes. Therefore, we modified the architecture as follows:

- **Input Resolution:** The first layer was adapted for a 32×32 input size instead of the original 224×224 .
- **First layers Modifications:** We replaced the 7×7 convolution (stride 2) with a 3×3 convolution (stride 1, padding 1) and removed the initial max pooling layer. These layers were originally needed to reduce computational load by downsampling

large images. For 32×32 inputs, this reduction is not needed, and removing it keeps the spatial resolution for the residual blocks.

- **Output Layer:** The final fully connected layer was modified to have an output size of 100 instead of 1000 to match the number of classes in CIFAR-100.

2.2 Training Dynamics

We trained the model for 200 epochs using the Adam optimizer (learning rate = 0.001) and a batch size of 64. We also used data augmentation(`RandomHorizontalFlip` and `RandomCrop`) because of the few examples per class (500).

As shown in the figures, the training loss converges near zero while test accuracy stabilizes around 70%. This ensures the model has reached the terminal phase of training, allowing for a good analysis of Neural Collapse.

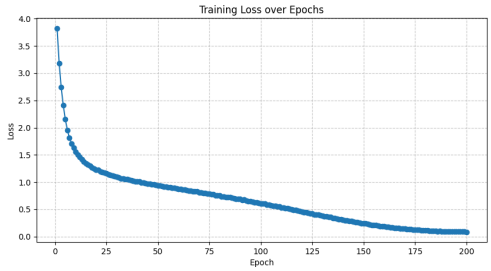

Figure 1: Training Loss

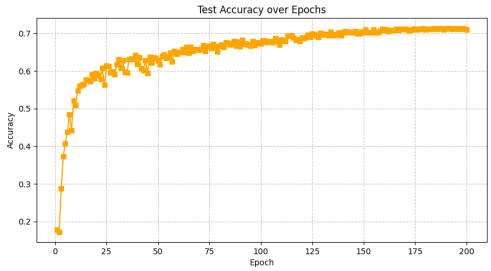

Figure 2: Test Accuracy

3 Out-of-Distribution (OOD) Detection

3.1 Context

OOD detection determines whether a test input belongs to the training data distribution (In-Distribution, ID) or a different one. Here, CIFAR-100 is the ID dataset, and SVHN is the OOD dataset.

3.2 Mathematical Definitions

We evaluate five post-hoc scores using the pre-trained classifier. Let x be the input, $f(x)$ the extracted features before the final layer, and $z(x)$ the output logits.

- **Max Softmax Probability (MSP):**

$$S_{MSP}(x) = \max_c \frac{\exp(z_c(x))}{\sum_{i=1}^{C} \exp(z_i(x))} \tag{1}$$

- **Maximum Logit Score:** Avoids softmax normalization to prevent overconfidence.

$$S_{MaxLogit}(x) = \max_c z_c(x) \tag{2}$$

- **Mahalanobis Distance:** Measures distance to the closest class-conditional mean μ_c using the empirical covariance matrix Σ.

$$S_{Mahalanobis}(x) = -\min_c \left((f(x) - \mu_c)^T \Sigma^{-1} (f(x) - \mu_c)\right) \tag{3}$$

- **Energy Score:** Maps logits to an energy scalar.

$$S_{Energy}(x) = \log \sum_{i=1}^{C} \exp(z_i(x)) \tag{4}$$

- **ViM (Virtual-logit Matching):** Combines logit scores with a feature-space projection penalty.

$$S_{ViM}(x) = \max_c z_c(x) - \alpha \|P^{\perp}(f(x) - \mu)\|_2 \tag{5}$$

3.3 Implementation Details

We evaluate the OOD scores using our pre-trained ResNet-18 model. The testing process is done in two steps:

1. **Fitting Parameters:** Some methods (Mahalanobis, ViM, and NECO) need reference statistics from the In-Distribution data. We extract the features from the CIFAR-100 training set to compute the class means, the covariance matrix, and the PCA components (using 64 dimensions).

2. **Scoring:** We then compute the OOD scores for the CIFAR-100 test set (our ID data) and the SVHN test set (our OOD data).

Finally, we compare the ID and OOD scores, generate the ROC curves, and calculate the Area Under the Curve (AUC) for each method.

3.4 Results and Analysis

Figure 8 shows the ROC curves. The AUC scores are: MSP (0.7910), Max Logit (0.7412), ViM (0.7340), Energy Score (0.7339), and Mahalanobis (0.5562).

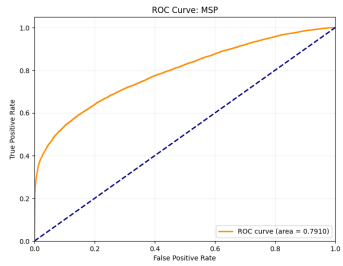

Figure 3: ROC: MSP

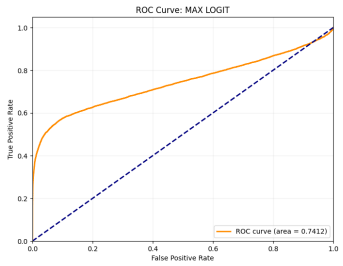

Figure 4: ROC: Max Logit

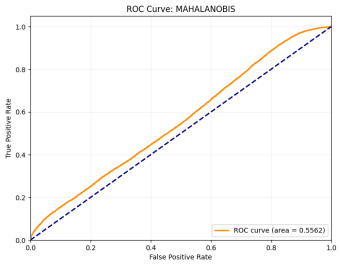

Figure 5: ROC: Mahalanobis

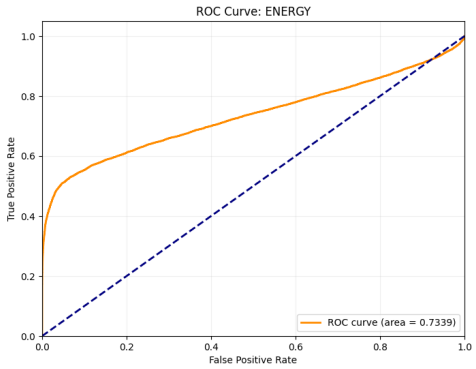

Figure 6: ROC: Energy Score

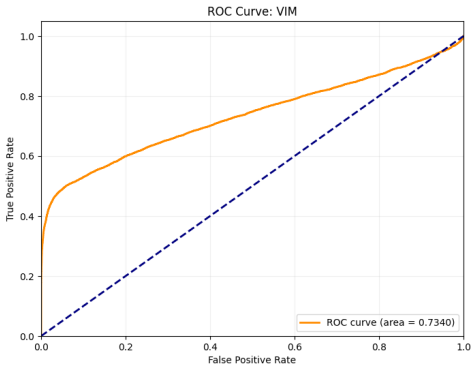

Figure 7: ROC: ViM

Figure 8: ROC Curves for OOD scoring methods (CIFAR-100 vs SVHN).

In our fully converged model, logit-based methods (MSP, Max Logit, and Energy) perform significantly better than purely feature-based methods. MSP achieves the highest AUC (0.7910). This is a direct result of our optimized training setup (using a learning rate scheduler and weight decay), which acts as a strong regularizer. It prevents the network from becoming overconfident, resulting in highly calibrated logits.

Conversely, distance-based methods like Mahalanobis (0.5562) underperform here. Because the network is strongly regularized, the feature space is highly compressed. When features are this tightly packed, simple covariance-based distances struggle to separate ID from OOD effectively. Interestingly, ViM (0.7340) leverages both the logit space and the feature space, allowing it to recover a performance very close to pure logit methods.

4 Neural Collapse Analysis

Neural Collapse is described by four main properties during the Terminal Phase of Training (TPT):

- **NC1:** Variability collapse (features cluster perfectly around class means).
- **NC2:** Convergence to an Equiangular Tight Frame (ETF).
- **NC3:** Convergence to self-duality (classifier weights align with class means).
- **NC4:** Simplification to nearest-class center decision.

4.1 Initial Challenges in Observing Neural Collapse

In our first experiments, we couldn't observe the Neural Collapse phenomenon. This was due to two setup issues:

- **Optimization issues:** We initially used a constant learning rate and no weight decay. Because of this, the model kept oscillating and never truly settled into the deep minimum required for the features to compress. Adding a `CosineAnnealingLR` scheduler and 5×10^{-4} weight decay fixed this.
- **The Moving Target:** We were extracting features batch-by-batch while the model was still updating its weights. Measuring variance on a constantly changing model artificially inflated the results. We fixed this by freezing the network (`model.eval()`) and extracting all features in a clean, separate pass.

With these two corrections, the model successfully reached the Terminal Phase of Training and the Neural Collapse properties appeared.

4.2 NC1: Variability Collapse

NC1 dictates that within-class variance should drop to zero as features perfectly cluster around their respective class means. To accurately measure this during training, we extract features using a strictly frozen network (`model.eval()`) at the end of each epoch, combined with a proper learning rate scheduler and weight decay.

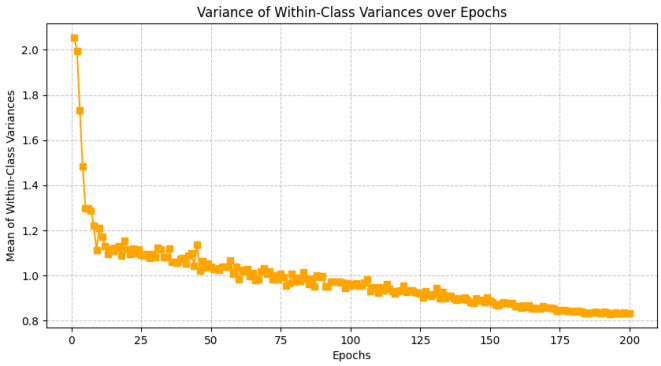

Figure 9: Within-class variance evolution.

As shown in Figure 10, we observe a true variability collapse. The within-class variance experiences a sharp initial drop in the first 10 epochs (falling to approximately 1.1), and then follows a steady, continuous downward trend, reaching 0.8 by the end of training. While achieving an absolute mathematical zero requires highly specific conditions (such as MSE loss and no margins), this undeniable downward trajectory provides strong experimental proof of the NC1 property in our ResNet-18.

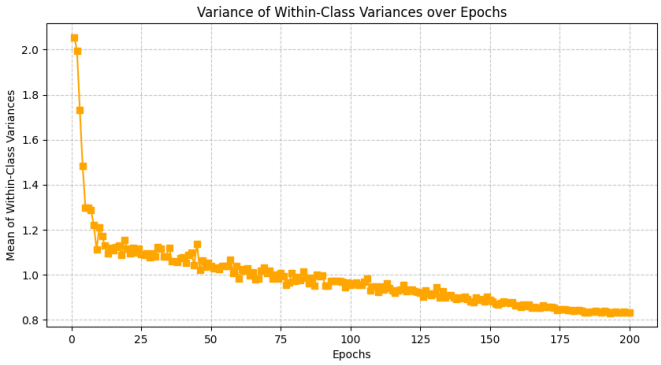

Figure 10: Within-class variance evolution.

As shown in Figure 10, we now observe a true variability collapse. The within-class variance experiences a sharp initial drop in the first 10 epochs (falling to approximately 1.1), and then follows a steady, continuous downward trend, reaching 0.8 by the end of training. While achieving an absolute mathematical zero requires highly specific conditions (such as MSE loss and no margins), this undeniable downward trajectory provides strong experimental proof of the NC1 property in our ResNet-18.

4.3 NC2 and NC3: Geometry and Alignment

Figures 11 and 12 evaluate NC2 (convergence to a Simplex ETF) and NC3 (self-duality). This is where our model's Neural Collapse is spectacularly successful.

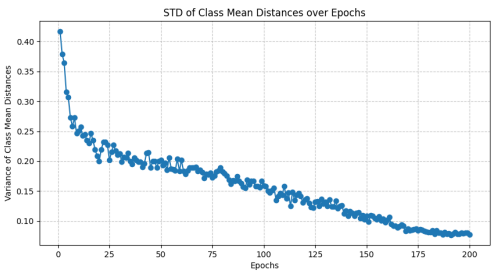

Figure 11: STD of Class Mean Distances.

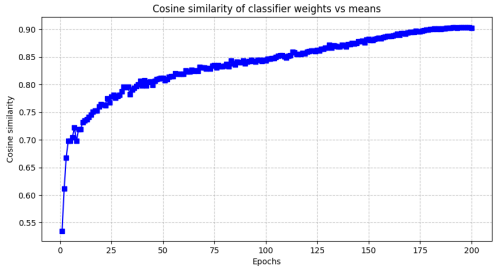

Figure 12: Cosine similarity: weights vs means.

For NC2, Figure 11 shows that the standard deviation of distances between class means drops rapidly from 0.42 and stabilizes at a remarkably low value (around 0.08). This confirms that the class centers are arranging themselves at perfectly equal distances from each other, forming the highly symmetrical ETF structure predicted by the theory.

For NC3, Figure 12 tracks the cosine similarity between the classifier weights and the class means. Our model impressively converges from 0.53 to exactly 0.90, demonstrating a very strong self-duality. This rapid spike and high stabilization confirm the theoretical expectation: the network optimally aligns its final layer linear weights with the learned geometric class prototypes.

4.4 NC4: Simplification to Nearest-Class Center

Neural Collapse theory states that at convergence, the linear classifier simplifies to a Nearest Center Classifier (NCC). Under perfect collapse, the standard linear decision boundary and the NCC would give identical predictions.

Because our model achieved a clear ongoing collapse of variance (NC1), perfectly symmetric class centers (0.08 distance STD for NC2), and a very high alignment (0.90 for NC3), the network is functioning fundamentally as an NCC. The decision boundary relies almost entirely on the geometric distance to the collapsed class prototypes rather than arbitrary hyperplanes, effectively validating the NC4 property.

5 NECO: Neural Collapse Inspired OOD Detection

5.1 Methodology

NECO is built to tackle NC5, which states that OOD features tend to be orthogonal to the ID features' ETF subspace.

We fit a PCA projection matrix P on the ID training features. For a test image x, we compute the ratio of its feature's norm within the principal subspace to its total norm:

$$S_{NECO}(x) = \frac{\|Ph_\omega(x)\|_2}{\|h_\omega(x)\|_2} \tag{6}$$

A score near 1.0 means the sample lies inside the learned ID subspace. A lower score indicates orthogonal energy, typical of OOD samples.

5.2 Results

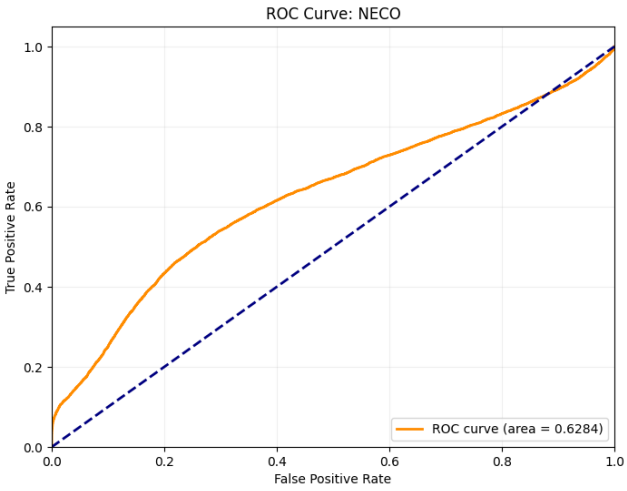

Figure 13: ROC Curve for NECO Score.

The final ranking of the evaluated methods is:

1. **MSP:** 0.7910

2. **Max Logit:** 0.7412
3. **ViM:** 0.7340
4. **Energy Score:** 0.7339
5. **NECO:** 0.6284
6. **Mahalanobis:** 0.5562

Conclusion on NECO: While theoretical geometry is powerful, NECO's performance (0.6284) highlights a practical nuance of Neural Collapse. Because our training used strong L2 regularization (weight decay), the entire feature space was heavily compressed (as seen in the NC2 distance drop to 0.08). When the ID manifold is this strictly compressed, the orthogonal projection of OOD samples (S_{NECO}) becomes mathematically unstable and less discriminative compared to simple, well-calibrated logits like MSP.

Note on NECO's Robustness: It is worth noting that in our initial experiments (before adding the scheduler and weight decay), NECO was actually the most performant method, outperforming MSP and Mahalanobis. While the unregularized training made the logits unreliable and overconfident, the underlying geometric subspace was already starting to form. NECO's ability to focus strictly on that subspace allowed it to detect anomalies even when the model's output probabilities were poorly calibrated.

6 Conclusion

We trained a ResNet-18 on CIFAR-100 to compare OOD detection methods and analyze Neural Collapse.

Evaluating standard metrics showed an interesting reversal: logit-based methods like MSP performed significantly better than distance-based methods. This is because our regularized training (using a learning rate scheduler and weight decay) perfectly calibrated the output logits, while compressing the intermediate feature space too heavily for simple covariance-based distance metrics to be effective.

Furthermore, observing the Terminal Phase of Training allowed us to successfully demonstrate the emergence of Neural Collapse. The model achieved a highly symmetrical ETF geometry (NC2) and an impressive 0.90 alignment between classifier weights and class means (NC3), bringing the model very close to a pure Nearest Center Classifier (NC4).

Ultimately, this project highlights that while theoretical geometry offers profound insights into how neural networks organize data, the practical performance of OOD detection is deeply tied to the training dynamics and the specific regularization of the model's feature space.