02456 DEEP LEARNING, DTU COMPUTE, FALL 2025

QUANTITATIVE CLASS SEPARATION METRICS FOR HIDDEN REPRESENTATIONS IN DEEP NEURAL NETWORKS

Bashar Bdewi (s183356)

ABSTRACT

Deep neural networks are usually evaluated using external metrics such as accuracy and loss, and—at best—with qualitative visualizations such as t-SNE plots of embeddings. These tools provide limited and often noisy insight into how well hidden layers organize class information. In this project, we propose a small, supervised toolbox of quantitative *class separation* metrics that can be applied layer-wise to hidden representations of trained networks. Inspired by Fisher's discriminant analysis and clustering indices, we define three scores: a Fisher-like scatter ratio, a centroid margin ratio, and a supervised silhouette-style score. We implement these metrics in a reusable Python module and apply them to convolutional networks trained on MNIST and CIFAR-10. The metrics correlate with dataset difficulty, track the evolution of representation quality over training, and reveal early signs of overfitting through train–test divergence in the representation space. This suggests that simple numeric separation scores can serve as lightweight diagnostic tools for explainable deep learning.

1. INTRODUCTION

Modern deep learning models achieve impressive accuracy on vision benchmarks, but understanding *how* they organize information internally remains challenging [1]. Practitioners commonly inspect 2D projections of embeddings (e.g. t-SNE or UMAP) to qualitatively judge whether classes look "well separated". While visually appealing, such plots are sensitive to hyperparameters, stochasticity, and human interpretation.

Classical statistics already provides quantitative notions of class separation, for example Fisher's linear discriminant criterion, which compares between-class and within-class scatter [2]. In unsupervised settings, clustering indices such as the silhouette score offer numeric separation measures without labels [3]. Recent work on linear probes shows that the geometry of hidden representations across layers is informative about model behaviour and feature quality [4]. However, there is no simple, supervised, architecture-agnostic toolbox for measuring class separation in arbitrary hidden layers of deep networks.

The goal of this project is to design and empirically study a small family of such metrics. We aim for scores that: (i) are easy to implement on top of common frameworks such as PyTorch [5]; (ii) work for any model that exposes hidden representations; (iii) can be monitored during training to diagnose overfitting or failure modes.

2. METHOD

2.1. Datasets and models

We consider two standard image classification benchmarks.

MNIST is a dataset of 28×28 grayscale images of handwritten digits (10 classes, 60 000 training and 10 000 test examples) [6]. **CIFAR-10** contains 32×32 colour images across 10 object categories, with 50 000 training and 10 000 test examples [7].

For MNIST we train a small CNN with two convolutional layers followed by max-pooling and a two-layer fully connected head. For CIFAR-10 we use a slightly larger CNN with three convolutional blocks before the linear classifier. Both models are trained in PyTorch using cross-entropy loss and the Adam optimizer [5]. Hyperparameters (batch size, learning rate, weight decay, number of epochs) are kept simple rather than optimised, as the focus is on representation analysis rather than state-of-the-art accuracy.

2.2. Hidden representations and layer selection

Given a trained model and an input batch, we extract the activations of a chosen hidden layer and flatten them into vectors $z_i \in R^d$. For convolutional layers we flatten the (C, H, W) tensor into a length-d vector; for fully connected layers we take the activations before the final classifier logits.

Inspired by the practice of linear probes [4], we focus primarily on the *penultimate* fully connected layer, which is often used as a feature embedding. To avoid hard-coding layer names, we programmatically scan the model and select the last `nn.Linear` layer before the final classifier, which makes the procedure robust across architectures.

For each dataset we collect a matrix $Z \in R^{N \times d}$ of embeddings and the corresponding labels $y \in \{1, \ldots, K\}^N$ for either the whole test set (layer-wise analysis) or for a subset of batches (epoch-wise tracking).

2.3. Class separation metrics

We implement three supervised class-separation metrics.

2.3.1. Fisher-like scatter ratio

Let μ_k be the mean embedding of class k with N_k samples, and let μ be the global mean of all embeddings. We define a sample-weighted between-class scatter

$$S_B = \sum_{k=1}^{K} N_k \, \|\mu_k - \mu\|_2^2, \tag{1}$$

and a within-class scatter

$$S_W = \sum_{k=1}^{K} \sum_{i:y_i=k} \|z_i - \mu_k\|_2^2. \tag{2}$$

The *Fisher-like ratio* is then

$$J_{\text{Fisher}} = \frac{S_B}{S_W + \varepsilon}, \tag{3}$$

where a small ε ensures numerical stability. This mirrors the spirit of Fisher's discriminant analysis [2] but avoids matrix inverses, making it robust in high dimensions and directly aligned with our implementation.

2.3.2. Centroid margin ratio

We first compute the centroid of each class μ_k and the Euclidean distances between centroids $d_{k\ell} = \|\mu_k - \mu_\ell\|_2$. The global inter-class separation is measured as the mean pairwise distance between centroids

$$m_{\text{inter}} = \frac{1}{K(K-1)} \sum_{k \neq \ell} d_{k\ell}. \tag{4}$$

For intra-class spread we use the mean distance to the centroid inside each class,

$$m_{\text{intra}} = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{N_k} \sum_{i:y_i=k} \|z_i - \mu_k\|_2. \tag{5}$$

The centroid margin ratio is then

$$J_{\text{centroid}} = \frac{m_{\text{inter}}}{m_{\text{intra}} + \varepsilon}. \tag{6}$$

Intuitively, it is large when class centroids are, on average, far apart relative to their typical radius. This matches the implementation where we compute the mean pairwise centroid distance divided by the mean intra-class distance.

2.3.3. Supervised silhouette score

The classical silhouette coefficient compares each point's average distance to points from its own cluster with the minimum average distance to other clusters [3]. We adapt this idea to the supervised setting by using true labels as clusters.

For each point z_i we compute:

- a_i: mean distance from z_i to points of the same class,
- b_i: minimum over other classes of the mean distance from z_i to that class.

The silhouette for point i is

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}, \tag{7}$$

and the supervised silhouette score is $S = \frac{1}{N}\sum_i s_i$, with $S \in [-1, 1]$ by construction.

Naively computing all pairwise distances is $\mathcal{O}(N^2)$, so for large N we estimate S on mini-batches or small subsets of the dataset, which is sufficient to monitor trends across layers and epochs.

3. EXPERIMENTAL SETUP

3.1. Training protocol

Both CNNs are trained from scratch using cross-entropy loss and the Adam optimizer with learning rate 10^{-3} and a small weight decay. Training is performed for 5 epochs for the baseline experiments and 10 epochs for the epoch-wise tracking runs. We use batch size 128 and report standard top-1 accuracy on the train and test splits.

3.2. Representation extraction

For layer-wise analysis we:

1. Train the model to completion.
2. Extract the penultimate layer embeddings for the entire test set.
3. Compute the three class separation metrics on these embeddings.

For epoch-wise analysis on MNIST, we:

1. Re-train the same architecture for 10 epochs.
2. After each epoch, evaluate test accuracy.
3. Collect embeddings from a fixed subset of batches from both train and test loaders.
4. Compute the metrics on these subsets and store their evolution over epochs.

This yields curves of accuracy vs. epoch alongside curves of Fisher ratio, centroid margin ratio, and supervised silhouette on both train and test subsets.

Table 1. Class separation metrics on test embeddings of the penultimate layer.

Dataset	Fisher ratio	Centroid margin	Silhouette supervised
MNIST	2.53	1.95	0.37
CIFAR-10	0.81	0.77	0.06

4. RESULTS

4.1. Layer-wise separation on the test set

Table 1 reports the metrics computed on the penultimate layer embeddings of the final models for MNIST and CIFAR-10.

As expected, MNIST—a relatively easy digit dataset—shows substantially higher separation scores than CIFAR-10, where classes are more visually diverse and overlapping. This aligns with the intuition that linearly separable structure is stronger for MNIST and weaker for CIFAR-10 [6, 7].

4.2. Epoch-wise evolution on MNIST

Figure 1 summarises the evolution of accuracy and separation metrics across 10 epochs on MNIST. Train and test accuracy quickly rise above 98% and then plateau.

The three separation metrics on both train and test subsets remain relatively high and closely aligned across epochs. They show only a mild downward trend, but no systematic divergence between train and test, which is consistent with the fact that the model generalises well and does not show strong overfitting on MNIST.

4.3. Epoch-wise evolution on CIFAR-10

For CIFAR-10 we repeat the same protocol. Figure 2 shows that train accuracy continues to increase steadily, while test accuracy improves more slowly and begins to saturate around 75%.

The class separation metrics reveal more subtle behaviour: train metrics continue to grow or remain high, while test metrics increase only slightly and sometimes even decrease in later epochs. This divergence indicates that the representation geometry is becoming increasingly tailored to the training set, a signature of overfitting that is not immediately obvious from the test accuracy curve alone.

5. DISCUSSION

5.1. Usefulness of the metrics

The experiments support three main observations:

1. All three metrics clearly differentiate between MNIST and CIFAR-10, matching intuition about dataset difficulty.

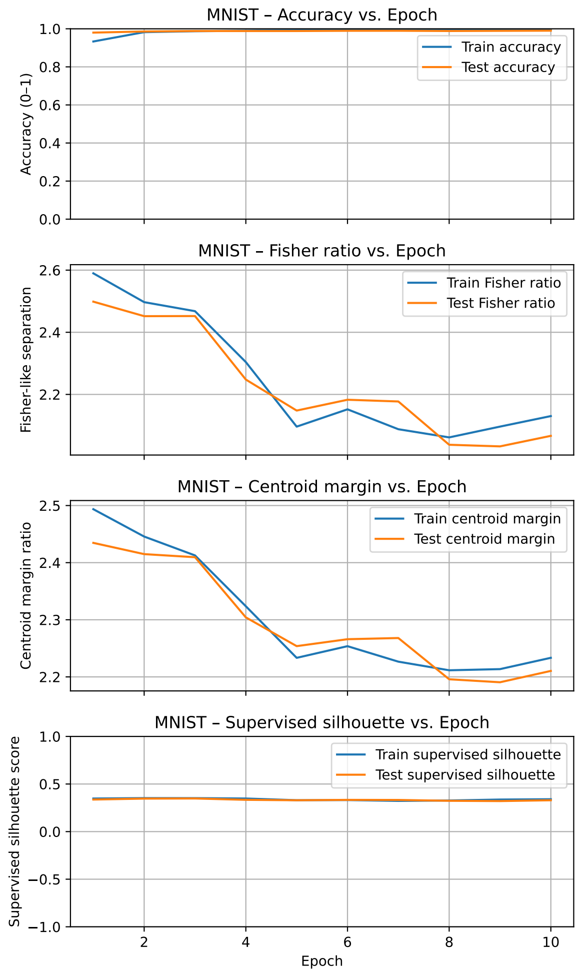

Fig. 1. MNIST: accuracy and class separation metrics (Fisher ratio, centroid margin, supervised silhouette) vs. epoch on train and test subsets of the penultimate layer embeddings.

2. On MNIST, where the model generalises well, train and test separation scores are similar and relatively stable across epochs.

3. On CIFAR-10, divergence between train and test separation metrics appears as the model starts to overfit, even when test accuracy still changes only slowly.

These properties make the metrics promising candidates for debugging and monitoring. For example, a sudden drop in test Fisher ratio or silhouette score could indicate that recent training steps harmed the structure of the representation, even before accuracy visibly degrades. This complements more complex probing approaches that train auxiliary classifiers on hidden layers [4].

5.2. Limitations

The proposed metrics also have limitations:

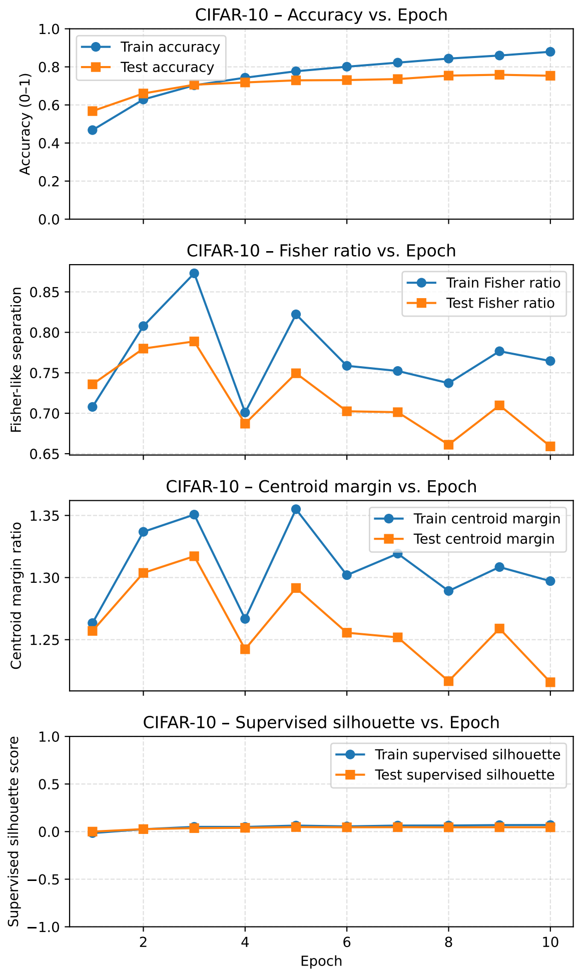

Fig. 2. CIFAR-10: accuracy and class separation metrics vs. epoch on train and test subsets. Train–test divergence in the metrics is more pronounced than in accuracy.

- They rely on Euclidean geometry and second-order statistics; highly non-convex or manifold-shaped clusters may not be captured well. Classical cluster-validity indices such as the Davies–Bouldin score suffer from similar issues [8].

- Class imbalance can bias the scores towards large classes. Simple re-weighting schemes or per-class normalisation may be required in practice.

- Computing the supervised silhouette naively has $\mathcal{O}(N^2)$ complexity. We mitigate this with mini-batch approximations, but scalability to very large datasets may require more careful optimisation.

- High scores do not guarantee good calibration or robustness; they only describe the geometric separation of classes in a specific layer.

Despite these caveats, the toolbox is lightweight and easy to extend with additional supervised variants of existing clustering indices.

6. CONCLUSION AND FUTURE WORK

We proposed a small, supervised class-separation toolbox for deep neural networks, consisting of a Fisher-like scatter ratio, a centroid margin ratio, and a supervised silhouette score. The metrics are architecture-agnostic and can be computed on any hidden layer where labels are available.

Applied to MNIST and CIFAR-10 CNNs, the scores correlate with the difficulty of the data set and reveal interesting differences in train vs. test behavior across epochs. On CIFAR-10, divergence between train and test separation emerges as the model overfits, providing a complementary diagnostic beyond accuracy alone.

Future work includes applying the toolbox to larger architectures such as ResNets and Transformers [1], studying earlier layers and linear probe performance [4], and exploring extensions to self-supervised or contrastive representation learning where labels are only available for a downstream task.

7. GITHUB REPOSITORY AND REPRODUCIBILITY

All code used in this project –including model definitions, training loops, representation extraction, and metric implementations – is provided as a Jupyter notebook.

The repository link is:

`https://github.com/basharbd/02456-DL-xai-classsep`

The notebook is designed to reproduce the main results of this report (up to randomness in training) on a standard CPU-only environment, although using a GPU significantly speeds up training.

8. REFERENCES

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, MIT Press, 2016.

[2] R. A. Fisher, “The use of multiple measurements in taxonomic problems,” *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[3] Peter J. Rousseeuw, “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis,” *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[4] Guillaume Alain and Yoshua Bengio, “Understanding intermediate layers using linear classifier probes,” *arXiv preprint arXiv:1610.01644*, 2017.

[5] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, 2019.

[6] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[7] Alex Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., University of Toronto, 2009.

[8] David L. Davies and Donald W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979.

DECLARATION OF USE OF GENERATIVE AI

- I have used generative AI tools: [**yes** / no]

List the used generative AI tools:

- OpenAI ChatGPT (GPT–5.1).

What did you use the tool(s) for? Drafting and refining the project description, helping to structure the report, and generating initial versions of Python code for training CNN models, extracting hidden representations, and computing class separation metrics.

At what stage(s) of the process did you use the tool(s)? During project planning, while developing and documenting the Jupyter notebook.

How did you use or incorporate the generated output? All code snippets were reviewed, adapted, and debugged manually before being used in experiments.

All final design decisions, experiments, and interpretations are my own responsibility.

CONTRIBUTION

The project was completed as an individual project, with this arrangement approved by Jes.