

Mathematical Formulations of Evaluation Metrics

1 Evaluation Metrics

To evaluate the state of the training, we employ a suite of metrics monitoring distinct aspects related to classical few-shot recognition.

1. **Recall and Precision:** We explicitly monitor raw Precision and Recall to maintain a fundamental understanding of the classifier’s performance. While derived metrics provide specific insights, tracking these raw values allows us to observe the immediate trade-off between the system’s exactness (the absence of false positives) and its completeness (the absence of false negatives) without the abstraction of complex ranking or thresholding layers.
2. **F1 Score:** This metric serves as the harmonic mean of Precision and Recall. It provides a single, unified score that balances both concerns, ensuring that an increase in Precision does not come at a disproportionate cost to Recall.
3. **Recall at Fixed Precision (R@P):** this is the recall score obtained when the system threshold is tuned to achieve a specific precision percentage. This metric is used specifically to address the problem of false positives.
4. **Discriminant Ratio (J):** We employ this metric for cluster validation within the generated d -dimensional embedding space. J is defined as the ratio between inter-class variance (S_B) and intra-class variance (S_W). We aim to maximize J , as a high value indicates the model has learned a "generalized" representation: producing highly similar vectors for images of the same logo (low intra-class variation) while creating distinct vectors for logos of different classes (high inter-class variance).
5. **Mean Average Precision (mAP):** This metric is adopted to track the quality of the generated rankings. In a retrieval context, it is insufficient to merely identify the top-1 match; the system must effectively rank all relevant instances (e.g., the 2nd or 3rd matches). mAP is optimal for evaluating the precision of the ranking across the entire query set. We prioritize this metric in the evaluation part for a particular advantage:
 - **Threshold Independence:** The selection of a specific similarity threshold is often arbitrary and brittle. Unlike binary classification metrics, mAP summarizes performance over the entire recall spectrum, effectively averaging precision across all possible thresholds.

1.1 Summary of Metric Selection

We selected **Recall at Fixed Precision** to monitor accuracy and recall under the high-confidence constraints required by the few-shot recognition challenge. We employ the **Discriminant Ratio** for clustering validation; this allows us to understand the spatial organization of the embedding and diagnose "why" the model is performing well. For general ranking performance, we selected **mAP** as the primary, general-purpose metric to track the overall sanity of the model. We also include the **F1 Score** to ensure a balanced optimization between false positives and false negatives. Furthermore, we track raw **Precision and Recall** to validate the fundamental retrieval mechanics of the system.

2 Fundamental Definitions

To establish the evaluation framework, we first define the fundamental classification outcomes based on the retrieval of a query image:

- **True Positive (TP):** A relevant logo is correctly retrieved/classified.
- **False Positive (FP):** An irrelevant logo (or different brand) is incorrectly retrieved/classified as relevant.
- **False Negative (FN):** A relevant logo is missed (not retrieved) by the system.

From these, we derive the core metrics:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (1)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (2)$$

3 Similarity Measure

To evaluate the similarity between a query image embedding $\mathbf{q} \in \mathbb{R}^d$ and a support set image embedding $\mathbf{x} \in \mathbb{R}^d$, we utilize the **Cosine Similarity**. This metric is chosen to normalize magnitude variations caused by illumination changes.

$$\text{sim}(\mathbf{q}, \mathbf{x}) = \frac{\mathbf{q} \cdot \mathbf{x}}{\|\mathbf{q}\|_2 \|\mathbf{x}\|_2} = \frac{\sum_{i=1}^d q_i x_i}{\sqrt{\sum_{i=1}^d q_i^2} \sqrt{\sum_{i=1}^d x_i^2}} \quad (3)$$

4 Ranking Performance: Mean Average Precision (mAP)

To evaluate the quality of the retrieval ranking, we compute the Mean Average Precision. This metric penalizes the model if relevant logos are ranked lower than non-relevant ones.

Given a set of query images Q , the mAP is defined as:

$$\text{mAP} = \frac{1}{|Q|} \sum_{q \in Q} \text{AP}(q) \quad (4)$$

Where $\text{AP}(q)$ is the Average Precision for a specific query q , calculated as:

$$\text{AP}(q) = \frac{1}{N_q} \sum_{k=1}^M P(k) \times \mathbb{K}(k) \quad (5)$$

Where:

- N_q : Total number of relevant ground-truth items for query q .
- M : The total number of items in the retrieval gallery.
- $P(k)$: Precision at rank k .
- $\mathbb{K}(k)$: Binary indicator function (1 if the item at rank k is relevant, 0 otherwise).

5 Cluster Quality: Discriminant Ratio (J)

To validate the topological quality of the embedding space, we calculate the Discriminant Ratio J . This metric assesses the ratio between Inter-class Scatter (S_B) and Intra-class Scatter (S_W).

$$J = \frac{\text{Tr}(S_B)}{\text{Tr}(S_W)} \quad (6)$$

The scatter matrices are defined as follows:

$$S_W = \sum_{c=1}^C \sum_{i=1}^{n_c} (\mathbf{x}_i^{(c)} - \boldsymbol{\mu}_c)(\mathbf{x}_i^{(c)} - \boldsymbol{\mu}_c)^T \quad (7)$$

$$S_B = \sum_{c=1}^C n_c (\boldsymbol{\mu}_c - \boldsymbol{\mu})(\boldsymbol{\mu}_c - \boldsymbol{\mu})^T \quad (8)$$

Where:

- C : Total number of distinct logo classes.
- $\boldsymbol{\mu}_c$: The mean vector (centroid) of class c .
- $\boldsymbol{\mu}$: The global mean vector of the entire dataset.
- $\text{Tr}(\cdot)$: The trace operator.

6 Operational Metric: Recall at Fixed Precision

To assess the operational feasibility of the system under strict confidence constraints, we define Recall at Fixed Precision (R@P). Let π be the target precision (e.g., $\pi = 0.95$). We first identify the decision threshold τ^* such that:

$$\tau^* = \min\{\tau \in [0, 1] \mid \text{Precision}(\tau) \geq \pi\} \quad (9)$$

The Recall at this specific operating point is then calculated as:

$$\text{R@P}_\pi = \frac{\text{TP}(\tau^*)}{\text{TP}(\tau^*) + \text{FN}(\tau^*)} \quad (10)$$

Where:

- $\text{TP}(\tau^*)$: True Positives retrieved with similarity score $\geq \tau^*$.
- $\text{FN}(\tau^*)$: False Negatives (relevant items missed) with similarity score $< \tau^*$.

7 Balance Metric: F1 Score

To evaluate the harmonic balance between the precision and recall of the model, we compute the F1 Score. This provides a unified view of performance, particularly useful when classes are imbalanced.

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \quad (11)$$

8 Implementation in the Training Loop

To ensure the model converges towards a distinct few-shot representation, the defined metrics are implemented as functional blocks within the training and validation loops. Below, we formalize the implementation of the Discriminant Ratio and Mean Average Precision as computational functions taking specific tensor inputs and yielding scalar outputs.

8.1 Discriminant Ratio Implementation

In the training phase, the Discriminant Ratio (J) serves as a monitor for cluster compactness and separability. It is implemented as a function f_J operating on a mini-batch of embeddings.

Function Signature:

$$J_{\text{score}} = f_J(\mathbf{E}, \mathbf{y})$$

Inputs:

- $\mathbf{E} \in \mathbb{R}^{B \times d}$: A tensor representing the batch of d -dimensional embeddings, where B is the batch size.
- $\mathbf{y} \in \mathbb{Z}^B$: The corresponding class labels for the batch.

Algorithmic Process:

1. **Centroid Computation:** For each unique class c present in \mathbf{y} , compute the class mean $\boldsymbol{\mu}_c$:

$$\boldsymbol{\mu}_c = \frac{1}{|\{i : y_i = c\}|} \sum_{i: y_i = c} \mathbf{E}_i$$

2. **Global Mean:** Compute the mean over the entire batch $\boldsymbol{\mu} = \frac{1}{B} \sum_{i=1}^B \mathbf{E}_i$.

3. **Scatter Matrices:** The Within-Class Scatter matrix (S_W) and Between-Class Scatter matrix (S_B) are approximated using batch statistics:

$$S_W = \sum_c \sum_{i \in c} (\mathbf{E}_i - \boldsymbol{\mu}_c)(\mathbf{E}_i - \boldsymbol{\mu}_c)^T$$

$$S_B = \sum_c n_c (\boldsymbol{\mu}_c - \boldsymbol{\mu})(\boldsymbol{\mu}_c - \boldsymbol{\mu})^T$$

4. **Output Generation:** Compute the trace of both matrices to determine the scalar ratio:

$$\text{Output} = \frac{\text{Tr}(S_B)}{\text{Tr}(S_W) + \epsilon}$$

Note: A small ϵ is added for numerical stability.

8.2 Mean Average Precision (mAP) Implementation

Due to the computational cost of ranking, mAP is implemented as a validation callback, executed at the end of each training epoch rather than per iteration. It evaluates the model against a query set (Q) and a gallery set (G).

Function Signature:

$$\text{mAP}_{\text{val}} = f_{\text{mAP}}(\mathbf{Q}, \mathbf{G}, \mathbf{y}_q, \mathbf{y}_g)$$

Inputs:

- $\mathbf{Q} \in \mathbb{R}^{N_q \times d}$: Tensor of Query embeddings.
- $\mathbf{G} \in \mathbb{R}^{N_g \times d}$: Tensor of Gallery (Support) embeddings.
- $\mathbf{y}_q, \mathbf{y}_g$: Label vectors for query and gallery sets respectively.

Algorithmic Process:

1. **Distance Matrix Computation:** Compute the pairwise Cosine Similarity matrix $\mathbf{S} \in \mathbb{R}^{N_q \times N_g}$ via matrix multiplication:

$$\mathbf{S} = \frac{\mathbf{Q} \cdot \mathbf{G}^T}{\|\mathbf{Q}\| \cdot \|\mathbf{G}^T\|}$$

2. **Ranking:** For each query i (row in \mathbf{S}), sort the gallery indices based on similarity scores in descending order to obtain the ranking indices π_i .
3. **Relevance Mapping:** Construct a binary matrix $\mathbf{R} \in \{0, 1\}^{N_q \times N_g}$ where $R_{i,j} = 1$ if $\mathbf{y}_q^{(i)} == \mathbf{y}_g^{(\pi_i(j))}$, and 0 otherwise.
4. **Cumulative Precision:** Compute the precision at every rank k :

$$P_i(k) = \frac{\sum_{j=1}^k R_{i,j}}{k}$$

5. **Output Generation:** Calculate the mean over all queries:

$$\text{Output} = \frac{1}{N_q} \sum_{i=1}^{N_q} \left(\frac{1}{\sum R_{i,:}} \sum_{k=1}^{N_g} P_i(k) \cdot R_{i,k} \right)$$

9 Asymptotic Computational Analysis

We analyze the time complexity of the defined metrics to understand their impact on the training pipeline throughput. The analysis assumes standard matrix multiplication algorithms.

9.1 Discriminant Ratio (J) Complexity

The Discriminant Ratio is computed per mini-batch. Let B be the batch size and d be the embedding dimension.

1. **Centroid Computation:** Computing the mean vectors requires iterating over all B samples of dimension d .

$$T_{\text{mean}} = \mathcal{O}(B \cdot d)$$

2. **Scatter Matrices Construction:** The core computational cost lies in calculating the Within-Class (S_W) and Between-Class (S_B) scatter matrices. Both involve computing outer products of the form $(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T$, resulting in a $d \times d$ matrix for each of the B samples.

$$T_{\text{scatter}} = \mathcal{O}(B \cdot d^2)$$

3. **Trace Operation:** Computing the trace of a $d \times d$ matrix is linear with respect to the diagonal.

$$T_{\text{trace}} = \mathcal{O}(d)$$

Total Complexity:

$$T_J = \mathcal{O}(B \cdot d^2)$$

Analysis: The complexity is linear with respect to the batch size but quadratic with respect to the embedding dimension. For typical few-shot scenarios where d is moderate (e.g., 128 to 512), this overhead is negligible compared to the backpropagation pass. However, if d scales significantly, the scatter matrix computation becomes a bottleneck.

9.2 Mean Average Precision (mAP) Complexity

The mAP is computed on the validation set. Let N_q be the number of query images, N_g be the number of gallery images, and d be the embedding dimension.

1. **Similarity Matrix Computation:** Computing the cosine similarity between all queries and gallery items involves a matrix multiplication of $(N_q \times d)$ and $(d \times N_g)$.

$$T_{\text{sim}} = \mathcal{O}(N_q \cdot N_g \cdot d)$$

2. **Sorting/Ranking:** For each of the N_q queries, we must sort the N_g gallery items by similarity score. Using an efficient comparison sort (e.g., Quicksort or Mergesort):

$$T_{\text{sort}} = \mathcal{O}(N_q \cdot N_g \log N_g)$$

3. **AP Accumulation:** Traversing the sorted list to compute precision at rank k is linear with respect to the gallery size for each query.

$$T_{\text{accum}} = \mathcal{O}(N_q \cdot N_g)$$

Total Complexity:

$$T_{\text{mAP}} = \mathcal{O}(N_q \cdot N_g \cdot d + N_q \cdot N_g \log N_g)$$

Analysis: The mAP calculation scales quadratically with the dataset size (interactions between Query and Gallery sets). The term $N_q \cdot N_g \log N_g$ typically dominates when $N_g \gg d$. Consequently, mAP is computationally prohibitive to compute at every training step and is strictly restricted to epoch-end validation or periodic evaluation intervals.

10 Computational Feasibility on Google Colab (Free Tier)

We analyze the computational feasibility of the proposed metrics specifically within the constraints of the Google Colab Free Tier environment. This environment typically provisions an **NVIDIA T4 GPU** (16GB VRAM, 8.1 TFLOPS FP32) and approximately **12GB of System RAM**.

Using the statistical data from Table 1, we instantiate the analysis with concrete values from the LogoDet-3K dataset.

Dataset Parameters:

- Total Images (N): 158,652
- Total Classes (C): 3,000
- Embedding Dimension (d): 512 (Assumed based on ResNet backbone)

10.1 Discriminant Ratio Estimation (Per Batch)

The Discriminant Ratio is calculated iteratively during training. Assuming a standard mini-batch size of $B = 128$:

$$\text{Ops}_J \approx B \cdot d^2 = 128 \cdot (512)^2 = 128 \cdot 262,144 \approx 3.35 \times 10^7 \text{ FLOPs} \quad (12)$$

Feasibility Verdict: Negligible. With ≈ 33.5 million floating-point operations per batch, this calculation is trivial for an NVIDIA T4. The memory overhead is localized to the batch tensors and does not accumulate history. Consequently, J can be monitored continuously without affecting training throughput or risking memory exhaustion.

10.2 mAP Estimation (Validation Phase)

The mAP calculation requires pairwise comparisons between query and gallery sets. We analyze the feasibility of storing the resulting similarity matrix in the limited VRAM/RAM of the Colab environment.

10.2.1 Scenario A: Full Dataset Evaluation

Attempting to rank the entire dataset against itself ($N_q = N_g = 158,652$):

1. **Similarity Matrix Ops:** ≈ 12.8 TeraFLOPs.

2. **Memory Footprint:** The similarity matrix requires storing N^2 single-precision floats:

$$(1.58 \times 10^5)^2 \times 4 \text{ bytes} \approx 100.2 \text{ GB}$$

Conclusion: Infeasible. A requirement of ~ 100 GB far exceeds both the VRAM (16 GB) and System RAM (12 GB) available in the free tier. Attempting this calculation will result in an immediate runtime crash.

10.2.2 Scenario B: Realistic Validation Split (20%)

We assume a validation set $N_{val} \approx 31,730$ images. We treat these as queries against a gallery of the same size.

1. **Similarity Matrix Ops:**

$$(3.17 \times 10^4)^2 \cdot 512 \approx 5.15 \times 10^{11} \text{ FLOPs (0.5 TeraFLOPs)}$$

2. **Memory Footprint:**

$$(3.17 \times 10^4)^2 \times 4 \text{ bytes} \approx 4.02 \text{ GB}$$

Conclusion: Feasible with Management.

- **Compute:** 0.5 TeraFLOPs requires < 0.1 seconds on a T4 GPU, making the time cost negligible.
- **Memory Constraint:** The 4.02 GB requirement is significant. During training, the model weights, optimizer states, and gradient cache may occupy 10 – 12 GB of the 16 GB VRAM. Allocating an additional 4 GB for the similarity matrix poses a high risk of `CUDA Out Of Memory` errors.
- **Mitigation Strategy:** To ensure stability on Colab Free, the validation step must implement **CPU offloading**. The embeddings should be detached and moved to System RAM before computing the similarity matrix. Since System RAM (≈ 12 GB) is generally less utilized than VRAM during training, the 4 GB matrix fits comfortably there, trading a small amount of PCIe transfer latency for stability.