Relation Network (RN)

Objective

- Learn to classify new/unseen examples with very few labeled samples.
- Instead of memorizing, the model learns how to compare using deep learning.

Few-Shot Learning Setup

- **C-way K-shot:**
  - C = number of classes per episode.
  - K = number of labeled samples per class.
- Two sets per episode:
  - **Support Set:** Contains $C \times K$ labeled samples.
  - **Query Set:** Contains unlabeled examples to classify.

Model Components

1. Embedding Module (f_ϕ)

- A CNN processes each image (support or query) into a feature map or embedding.
- All images go through the same embedding function, ensuring a shared representation space.

2. Pairwise Comparison Setup

- For each query image, we compare it against every support image.
- Embeddings from both the query and support images are concatenated:

$$C(f_\phi(x_i), f_\phi(x_j))$$

3. Relation Module (g_ϕ)

- Accepts concatenated embeddings of (query, support) pairs.
- Outputs a relation score $r_{i,j} \in [0, 1]$ indicating the similarity or closeness between the query and the support image.
- Implemented as a small CNN or MLP.
- For each query, we obtain a vector of scores — one for each class in the support set.

4. Prediction with One-Hot Labels

- A one-hot label vector represents the true class of the query (e.g., $[0, 1, 0, 0, 0]$ for class 2 in 5-way classification).
- The highest relation score among all support classes is selected as the predicted class.

Training in Detail (Episodic Learning)

- Episodic training mimics few-shot testing setup:
  - Sample C classes per episode.
  - Select K support samples/class.
  - Create queries from same classes.
- For each query image x_i:
  1. Compare it with each support sample x_j.
  2. Get embeddings: $f_\phi(x_i)$, $f_\phi(x_j)$.
  3. Concatenate: $[f_\phi(x_i), f_\phi(x_j)] \rightarrow g_\phi \rightarrow$ relation score.
  4. Repeat for all support-query pairs.
- Output: One relation score per support class.
- Pick the class with the highest score as prediction.

Loss Function

Mean Squared Error (MSE) between predicted relation scores and ground-truth match indicators:

$$\mathcal{L} = \sum_{i=1}^{m} \sum_{j=1}^{n} (r_{i,j} - \mathbf{1}(y_i = y_j))^2$$

Where:

- $r_{i,j}$: Predicted relation score between query x_i and support class x_j.
- $\mathbf{1}(y_i = y_j)$: Indicator function, 1 if query and support share the same label, 0 otherwise.

Goal:

- Encourage high scores for correct matches.
- Penalize high scores for incorrect matches.

Optimization:

$$\phi, \varphi \leftarrow \arg\min_{\phi,\varphi} \mathcal{L}$$

Training Strategy – Episodic Learning

- Mimics real few-shot testing scenario during training:
  - Each episode has a support set (K examples/class) and a query set.
  - All query-support pairs are passed through the RN.
- For every query:
  - Relation scores are computed with each support class.
  - Class with highest score is predicted.

Disjoint Classes in Train vs. Test

- The classes in the training set and test set are disjoint.
- The model never sees test classes during training.
- Instead, it learns a generalizable way to compare images — a deep metric — that works across unseen categories.
- This ensures the model generalizes to new concepts by comparing relationships, not memorizing class-specific features.

Why It Works

- **Learns to Compare:** The model doesn't need to memorize all classes — it learns a general comparison function.
- **Flexible Matching:** Uses non-linear similarity (better than fixed metrics like cosine or Euclidean).
  The model doesn't use a fixed distance like Euclidean or cosine. Instead, it learns a task-specific distance function — the relation module g_ϕ — that is optimized end-to-end. This learned function is more flexible and expressive than hand-designed distance metrics.
- **End-to-End Differentiable:** Embedding + relation modules trained jointly.
- **Disjoint Labels:** Generalizes well even when test classes are different from training classes.
- **Meta-learning:** Learns the concept of "how to compare" instead of memorizing classes.
- **End-to-end training:** Embedding and relation functions are learned jointly.
- **Generalizes to new classes** — even those not seen during training.