**Objectif** : train a retrieval model in an end-to-end manner while exploiting multiple global descriptors without controlling diversity among the learners.

**Flexible and expandable** by GD, CNN backbone, loss and dataset.

We have a **CNN backbone network** and **2 modules** (combination of multiple global descriptors and auxiliary module that fine-tune with a classification loss).  
**Final loss** : ranking loss + classification loss

# 1. Backbone and Framework

**Backbone** : RestNet-50 without the down sampling operation between stage 3 and 4.  
To improve the performance because it contains richer information.   
**Input** : 224x224 image  
**Output** : 14x14 feature map

**12 Combinations** of GD possible with **SPoC** (S), **MAC** (M), and **GeM** (G).

**Data** : “For CUB200 and CARS196, cropped images with bounding box information are used. We follow the same training and test split as [9, 24, 60] for fair comparisons.”

## a. Implementation details

* Using **MXNet**
* **1536or768or512**-Dimensional embedding
* **Input during training** : resized to 252 x 252, cropped randomly to 224x224, and then flipped horizontally randomly.
* **Input during inference phase** : resize image to 224x224 only
* **Adam** **optimizer** with learning rate of 1e-4, step decay for scheduling LR.
* **Margin** for triplet loss = 0.1
* **Temperature** T softmax loss set to 0.5
* **Batch** size 128

# 2. Main module : Multiple Global Descriptors

Global descriptors generated by global pooling :

* Sum pooling of convolutions (**SPoC**) : large region[3],
* Maximum activation of convolutions (**MAC**) : focused region [53]
* Generalized mean pooling (**GeM**) [43]

## a. Pooling process

Global descriptor takes X as input and produces a vector f as output by **pooling process**. It gives :



We set :

* **SPoC** as f(s) when pc=1
* **MAC** as f(m) when pc->oo
* **GeM** as f(m?) for the rest of the cases, pc parameter 3 fixed here though

## b. Dimensionality reduction and Normalization

The output feature vector from the ith branch is obtained after **dimensionality reduction** (FC layer) and **normalization** (l2-normalization layer)



i : number of the branch

Wi : weight of the FC layer

F(a1) : global descriptor f(a1). ai=s -> **SPoC**. ai=m -> **MAC**. ai=g -> **GeM**

## c. Final feature vector

Combination of feature vectors using concatenation and l2-normalization



ai = s, m or g

Ranking loss : batch-hard triplet loss

# 3. Auxiliary Module (Classification Loss)

Softmax cross-entropy loss + label smoothing ?

It’ll fine tune the CNN backbone base on the first GD using a **classification loss**. Helps to maximize inter-class distance -> model train faster dans more stable.

Softmax loss defined as :



* N : batch size
* M : number of classes
* yi : identity label of ith input
* W : trainable weight
* b : bias
* f : first Global Descriptors
* T : temperature set to default value 1. Low temperature **allows larger gradient to challenging examples**. Thus, it’ll be helpful for intra-class compact and inter-class spread-out embedding.

Label smoothing improves generalization by “estimating the marginalized effect of a label-dropout during training”.   
Label smoothing and temperature scaling -> prevent over-fitting and learn better embedding.

Results of their experiment notes :

* Ranking loss + Auxiliary classification loss is better
  + Ranking loss : inter class ?
  + Classification loss : intra class ?
* Both tricks (label smoothing and temperature scaling) is better
* Concatenation of vectors is better than sum
* SG seems to be the best overall, don’t talk about why better results with two than three
* RestNet-50 is good