Similarity-Based Visual Search

Deep Hash Neural Network Implementation based on

Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks, CVPR 2015.

Yupan Huang (Sun Yat-SenUniversity) | 2017.4.14

Finding Similar Images

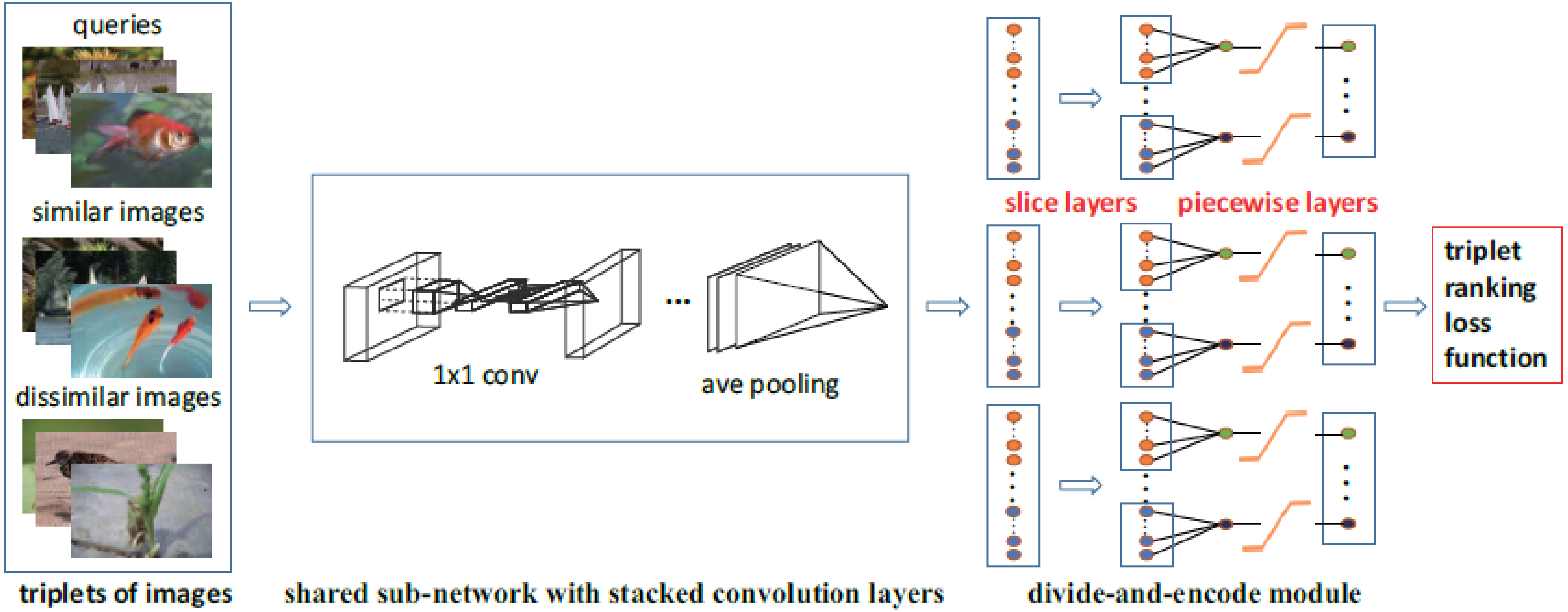
Assume I to be the image space. The goal of hash learning for images is to learn a mapping $F: I \rightarrow \{0,1\}^q$, such that an input image I can be encoded into a q -bit binary code $F(I)$, with the similarities of images being preserved.

Google Image Search Results

Motivation

- Extracting informative image features
  Deep Convolutional Neural Network:
  - Extract discriminative features depend on data
  - Build end-to-end relation between the raw image data and the binary hashing codes
- Learning effective approximate functions
  Hash Code:
  - Space-saving :high-dimensional features to lower dimensional space, compact bitwise representation
  - Speedup :binary pattern matching or Hamming distance measurement

Approach

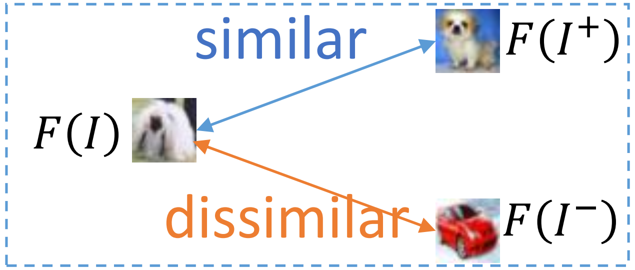


Three building blocks of DNNH

Part 1 Triplet Ranking Loss and Optimization

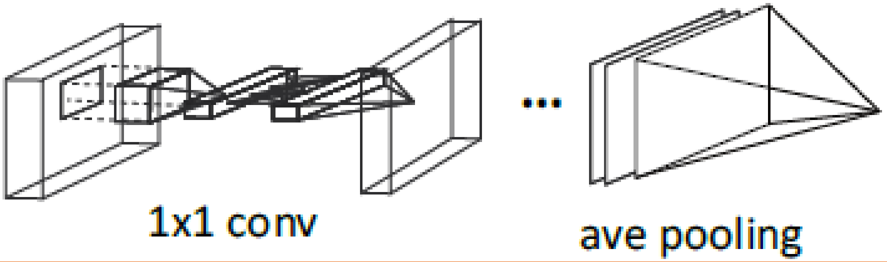
$$\hat{l}_{triplet}\left(F(I), F(I^+), F(I^-)\right)$$
$$= \max\left(0, 1 - \left(||F(I) - F(I^-)||_H - ||F(I) - F(I^+)||_H\right)\right)$$
$$s.t.\, F(I), F(I^+) \in \{0,1\}^q$$

Characterize that one image is more similar to the second image than to the third one



Relaxation

Forward inference: $f_w(x)$



Backward learning: $\nabla f_w(x)$

$$\hat{l}_{triplet}\left(F(I), F(I^+), F(I^-)\right)$$
$$= \max\left(0, ||F(I) - F(I^+)||_2^2 - ||F(I) - F(I^-)||_2^2 + 1\right)$$
$$s.t.\, F(I), F(I^+) \in [0,1]^q$$

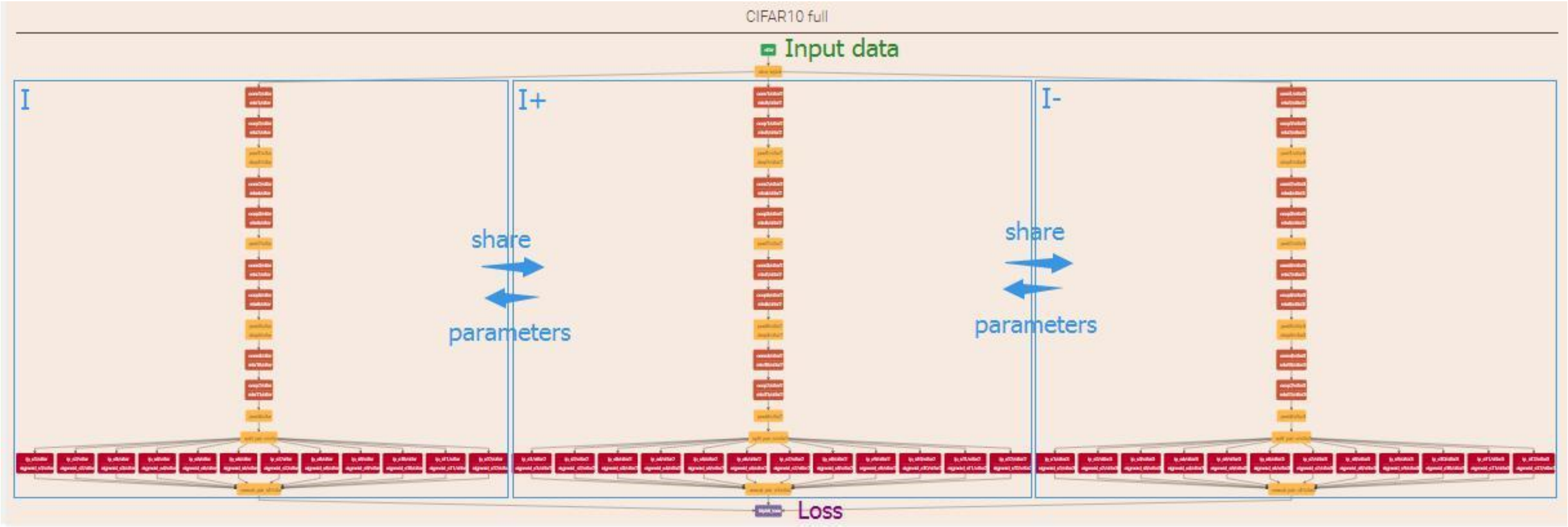
$$\frac{\partial l}{\partial b} = (2b^- - 2b^+) \times I_{||b-b^+||_2^2 - ||b-b^-||_2^2 + 1 > 0}$$

$$\frac{\partial l}{\partial b} = (2b^+ - 2b) \times I_{||b-b^+||_2^2 - ||b-b^-||_2^2 + 1 > 0}$$

$$\frac{\partial l}{\partial b} = (2b^- - 2b) \times I_{||b-b^+||_2^2 - ||b-b^-||_2^2 + 1 > 0}$$

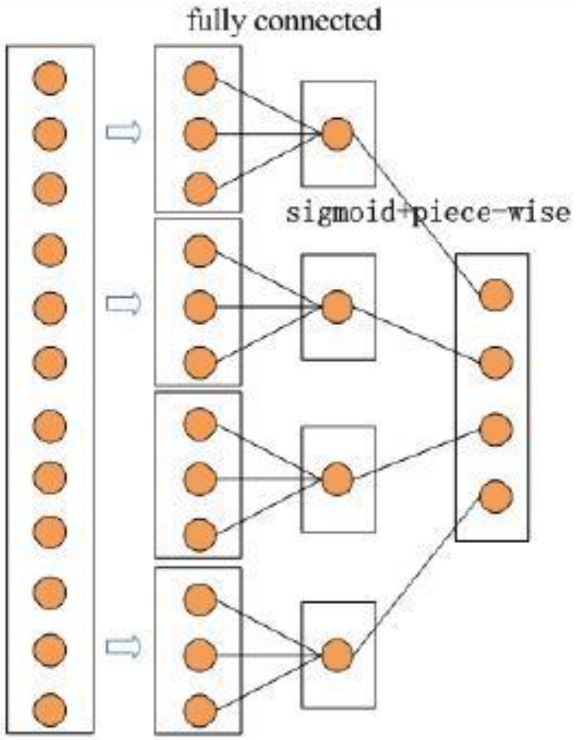
Part 2 Shared SubNetwork with Stacked Convolution Layers

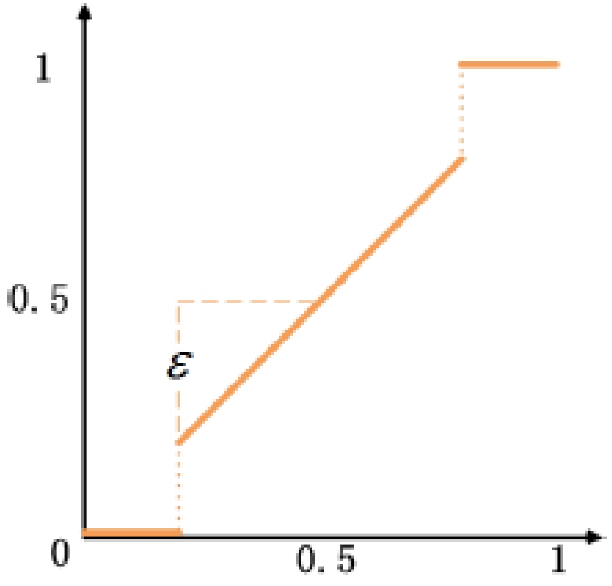
- Parameter sharing can significantly reduce the number of parameters in the whole architecture.



http://ethereon.github.io/netscope/#/gist/2f44d589dbe4355ad2f9374344527b6b

Part 3 Divide and Encode Module





- Separated slice of features: Reduce the redundancy among the hash bits
- Piece-wise threshold: encourage the output of binary hash bits:

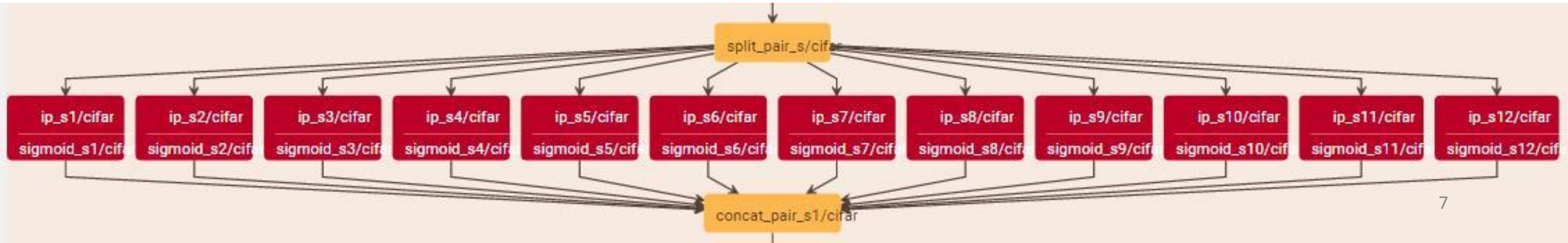
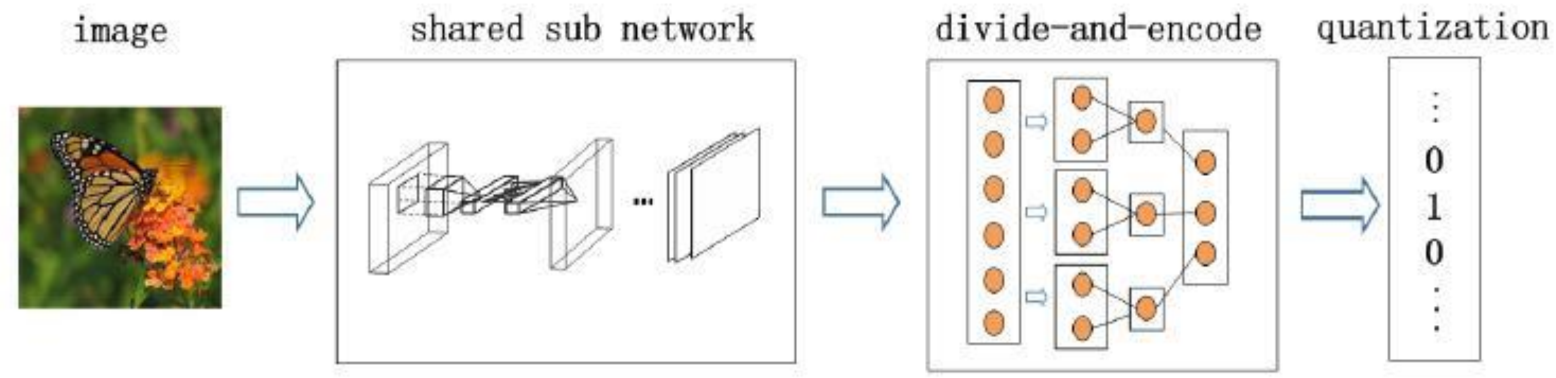
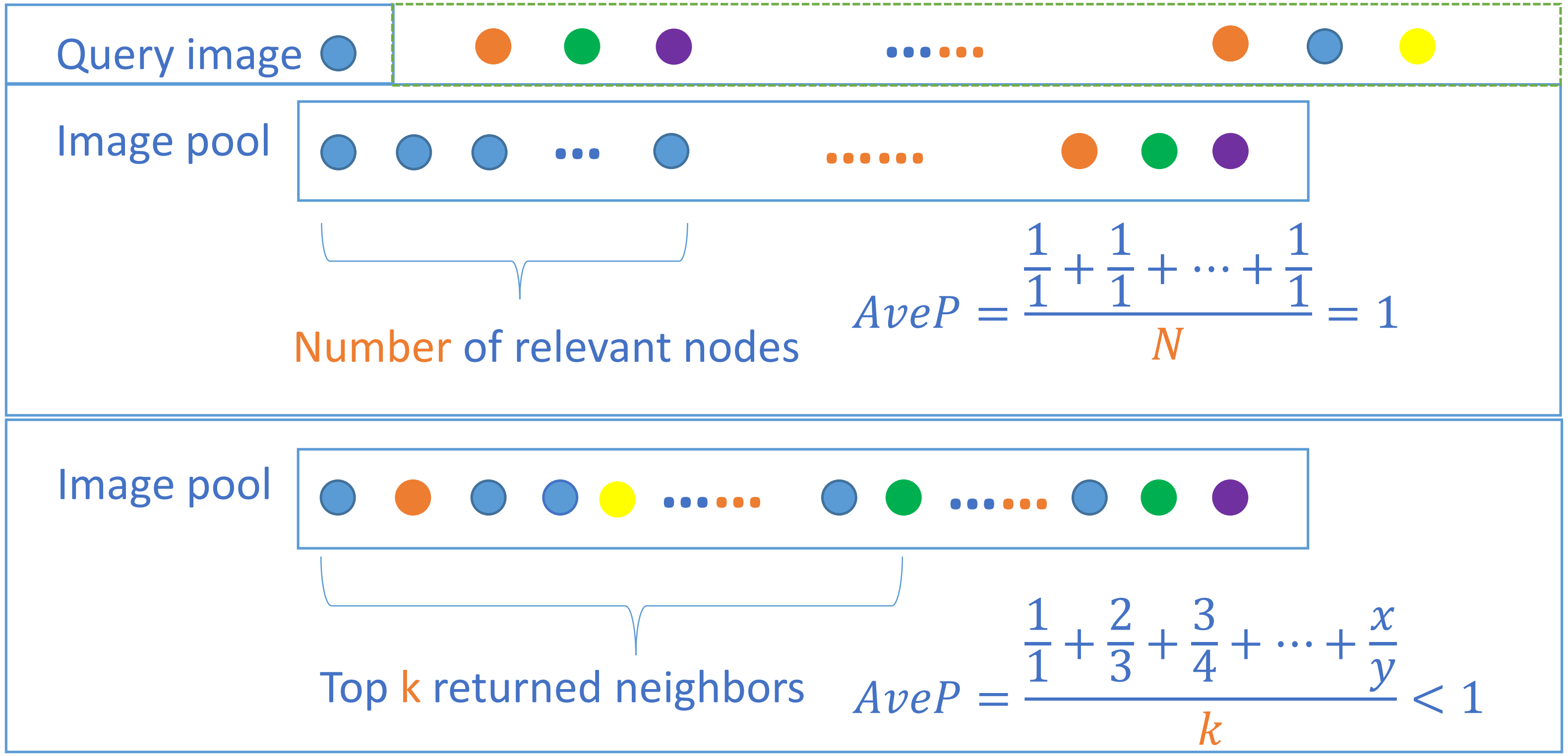


Image Retrieval and Model Evaluation



Given a query image, the retrieval list of images is produced by sorting the hamming distances between the query image and images in search pool.



$$AveP = \frac{\frac{1}{1}+\frac{1}{1}+\cdots+\frac{1}{1}}{N} = 1$$

$$AveP = \frac{\frac{1}{1}+\frac{2}{3}+\frac{3}{4}+\cdots+\frac{x}{y}}{k} < 1$$

$$MAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q}$$

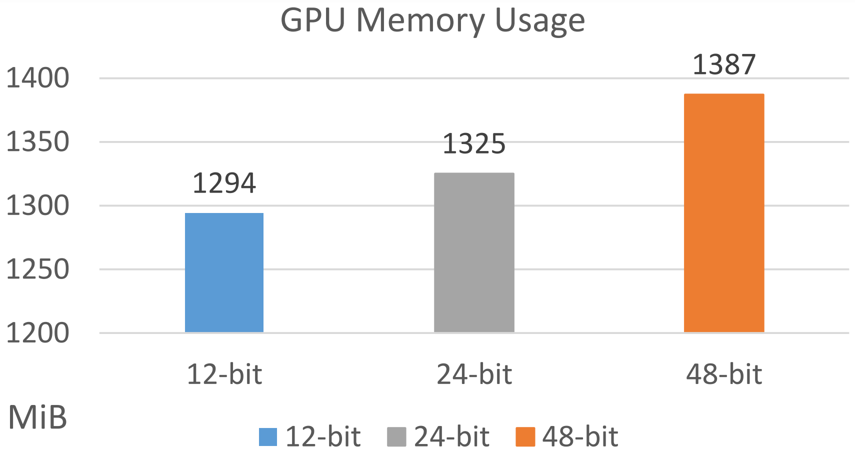
$$AveP = \frac{\sum_{k=1}^{N}(P(k) \times rel(k))}{\min(N,k)}$$

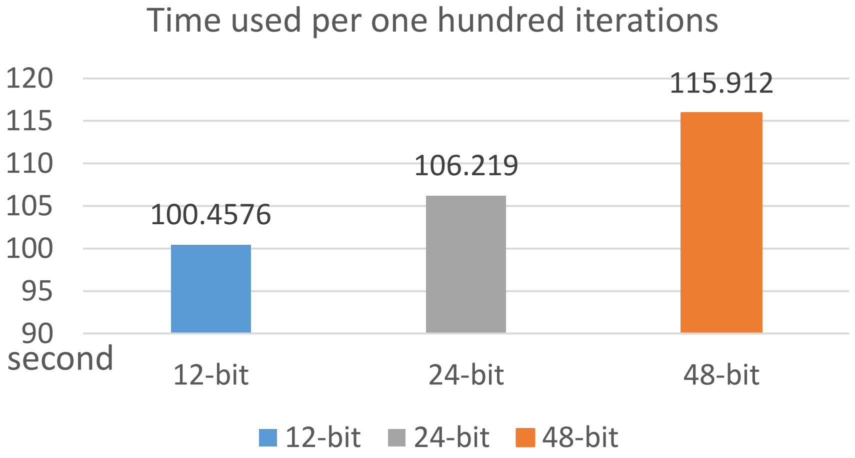
Mean Average Precision(MAP)

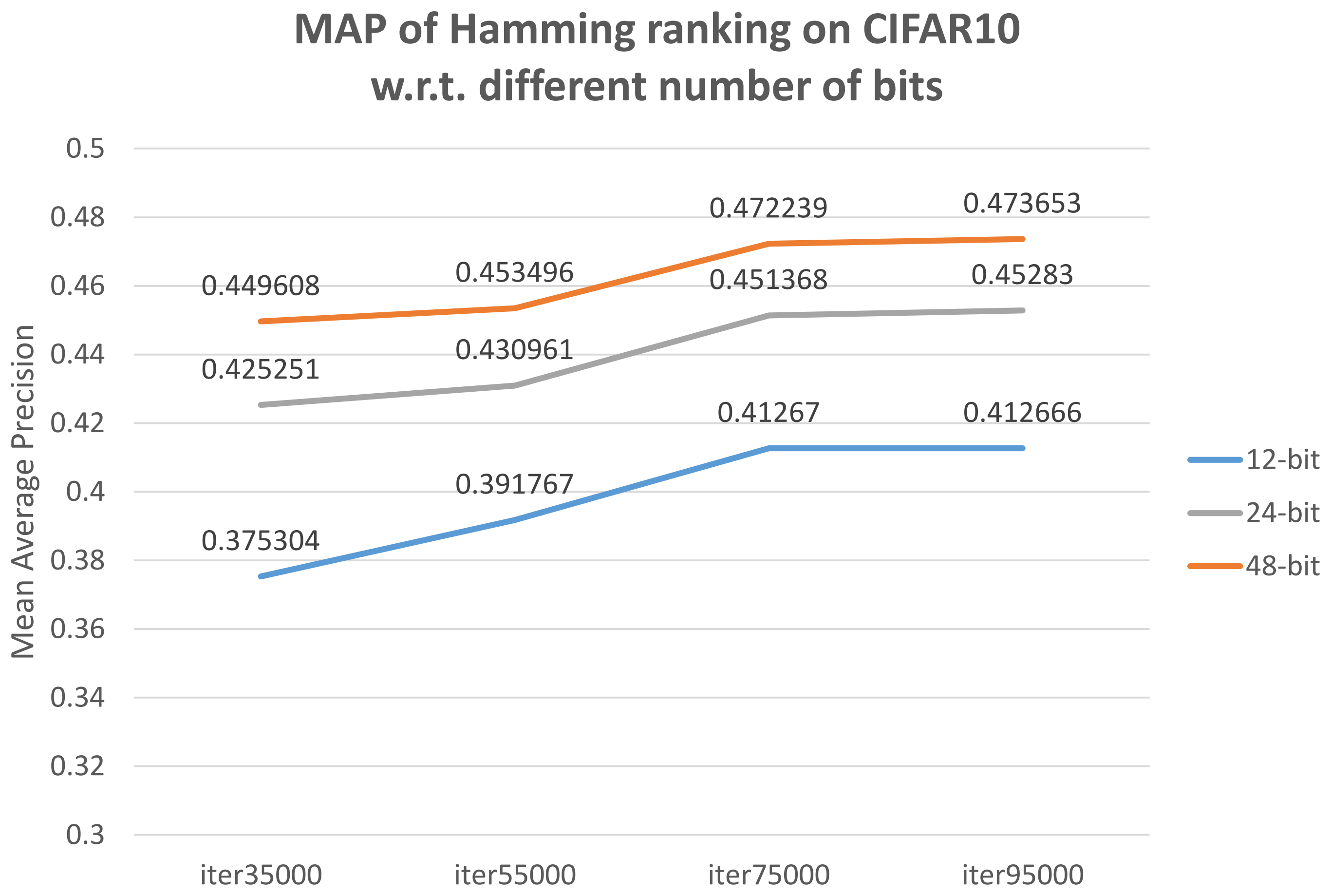
Implementation on Caffe framwork

- **Deploy**: Given the definition of loss layer, deploy the deep hashing pipeline on linux.
- **Train**: Write prototxt to define DNNH and bash files to execute for training on preprocessed triplet CIFAR-10 dataset.
- **Test**: Write prototxt to encode images into 12-bit, 32-bit and 48-bit seperately and bash files to execute for image retrieval.
- **Evaluate**: Implement the metric of mean average precision (mAP) for evaluation.

Results and Analysis







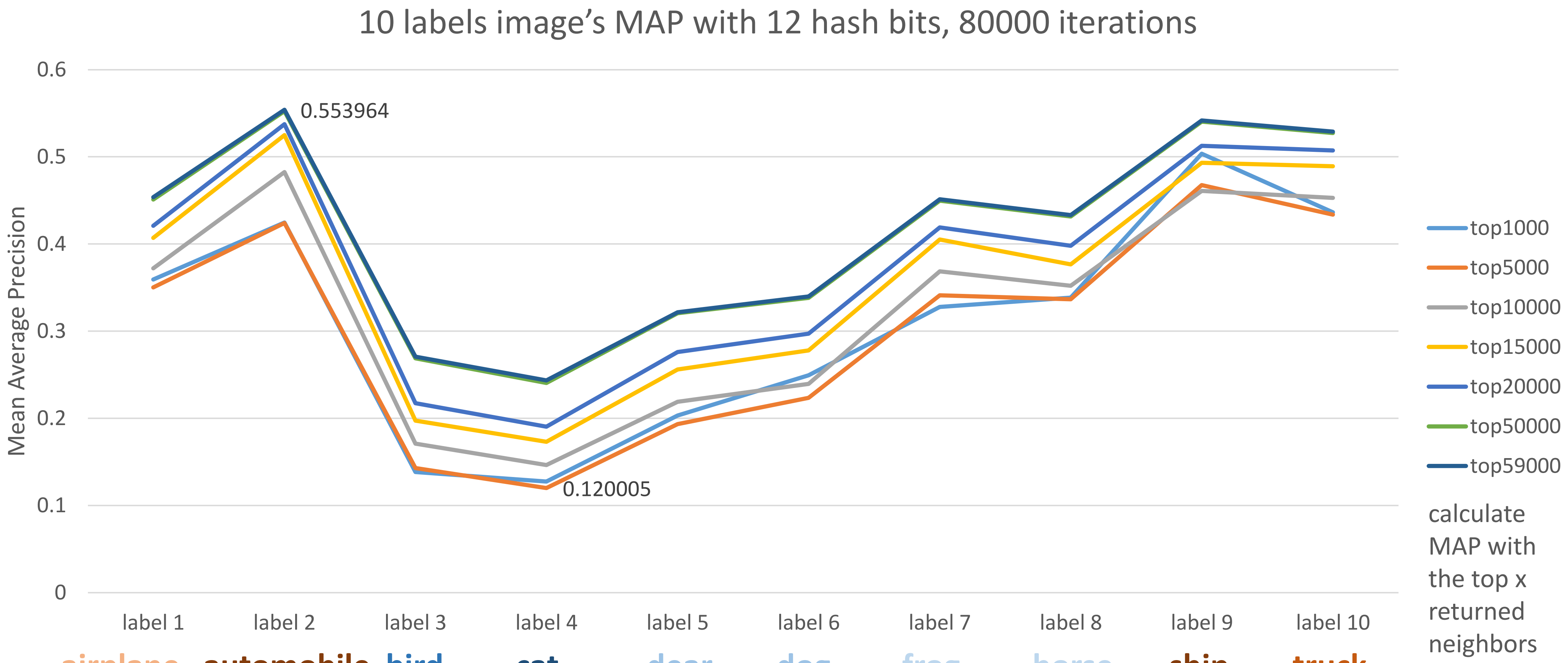
Shorter codes have the edge in space and time usage.

Longer codes have higher accuracy.

128-bit codes??
Shorter codes may be capable of presenting images and large bits may be overfitting.

Test on Tesla K80 GPU Card

Results and Analysis

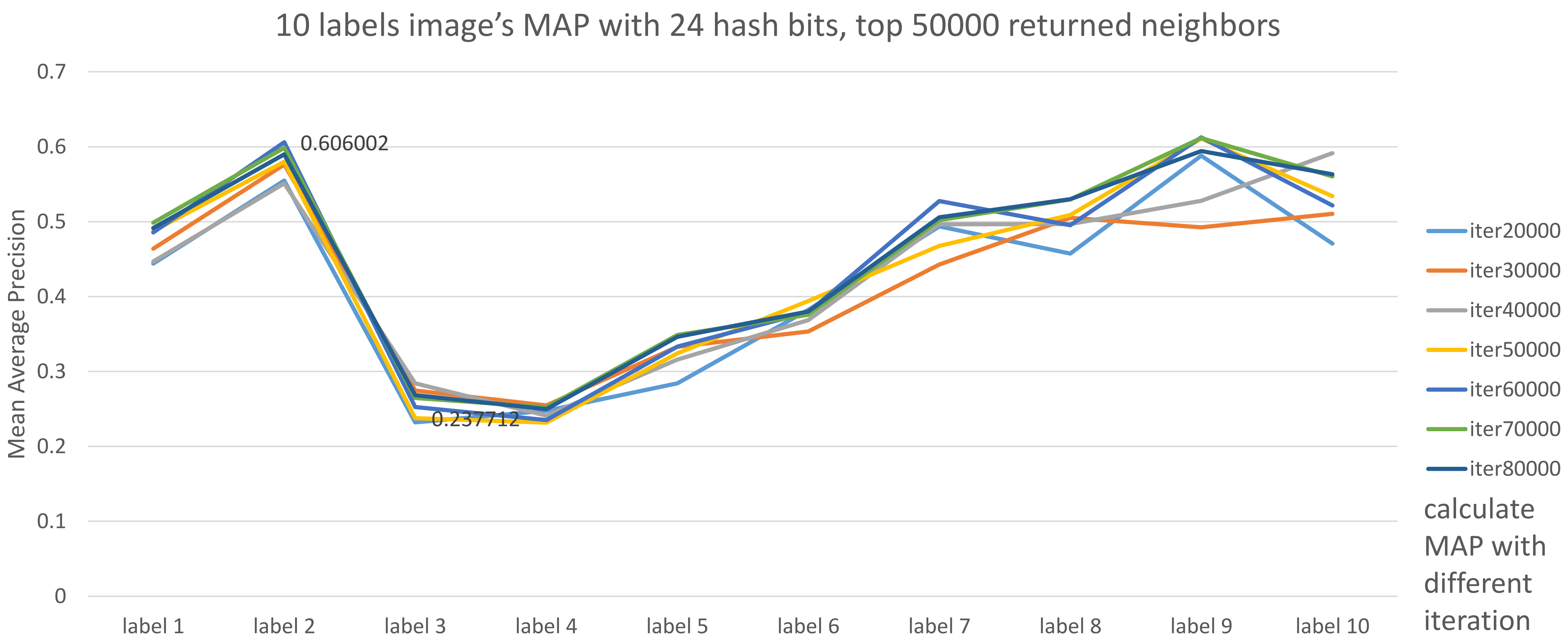


airplane automobile bird cat dear dog frog horse ship truck

Artifacts like automobile, ship, truck, airplane outperform animals like cat, bird, dear dog and so on.

Test on Tesla K80 GPU Card

Results and Analysis



calculate MAP with different iteration models

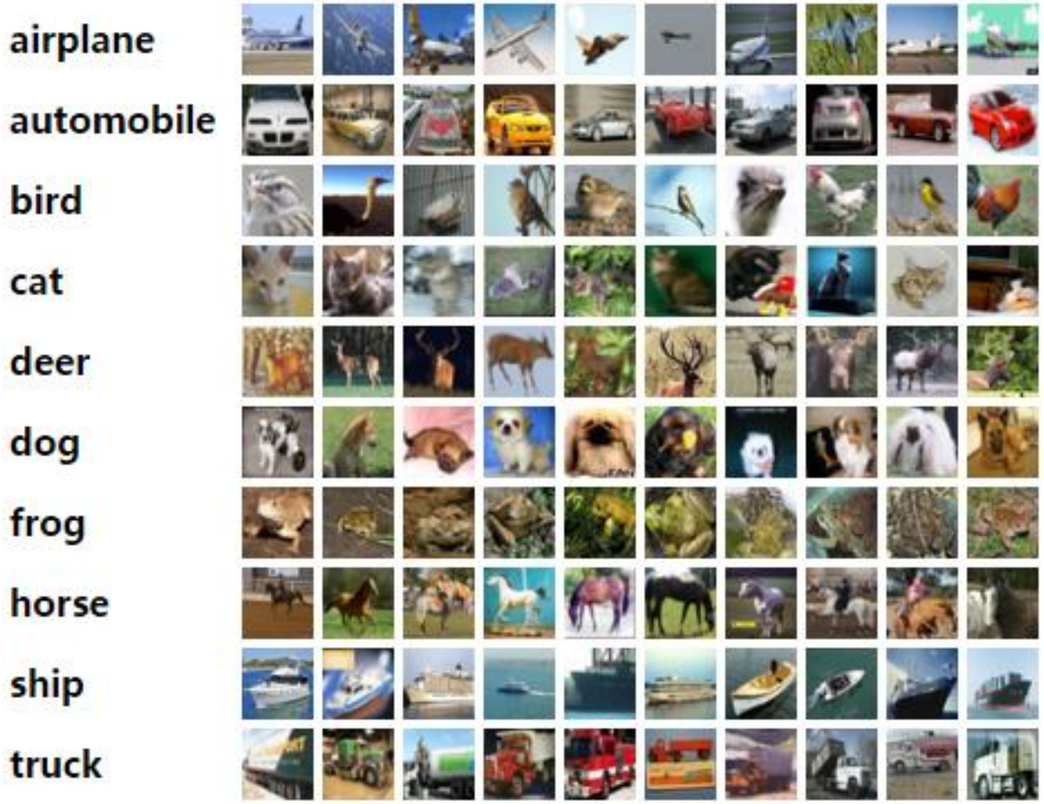
airplane automobile bird cat dear dog frog horse ship truck

Artifacts like automobile, ship, truck, airplane outperform animals like cat, bird, dear dog and so on.

Test on Tesla K80 GPU Card

Futher Improvements

- MAP closely relates to labels/classes in the CIFAR-10 dataset.
- Use different margin for different labels/classes.
- *Larger* margin for artifacts.
  Smaller margin for animals.



[The CIFAR-10 dataset]

$$\hat{l}_{triplet}\left(F(I), F(I^{+}), F(I^{-})\right)$$
$$= \max\left(0, margin - \left(||F(I) - F(I^{-})||_{H} - ||F(I) - F(I^{+})||_{H}\right)\right)$$
$$s.t. F(I), F(I^{+}) \in \{0,1\}^{q}$$

Futher Improvements

- Margin design
- Fine-tune on pre-trained deep models for better performance in both speed and precision.
- How about four images a group?
  For example: $\{F(I^{a,b}), F(I^{a}), F(I^{b}), F(I^{-})\}$
  $F(I^{a,b})$ is image with tag a and b, $F(I^{a})$ and $F(I^{b})$ are images with tag a and b respectively while $F(I^{-})$ neither has tag a or b.
  Aim to preserve semantic similarity as well.

Thank you!

For more details:

Source code in GITHUB.

Post in blog.