FPGA Implementation of In-Memory Search Macro with Range Encoding

Speaker : B11901027 王仁軒
Mentor : Rick Huang
Advisor: Prof. An-Yeu (Andy) Wu
Date : 2025/05/02

ACCESS IC LAB

1

Outline

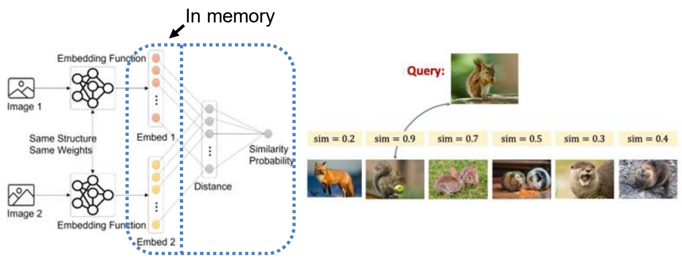
- Introduction to few-shot learning
- BORE range encoding scheme
- IMS system architecture
- Implementation on FPGA
- Result

- Embedding dimension reduction
- Analysis of PCA (Principle Component Analysis)
- Analysis of quantization method
- Modification of AutoEncoder method

2

Few-Shot Learning

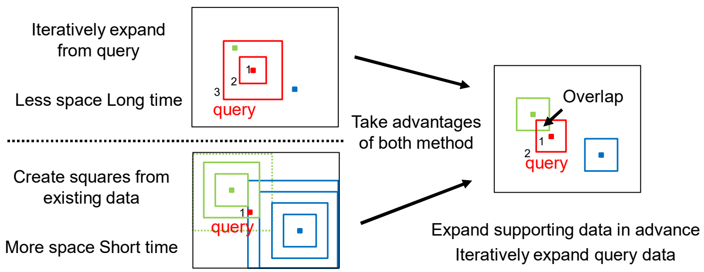
- In few-shot learning, we compare similarity between query and supporting data in memory
  - Process data with NN, store embeddings into memory
  - Compute similarity between query and supporting embeddings



3

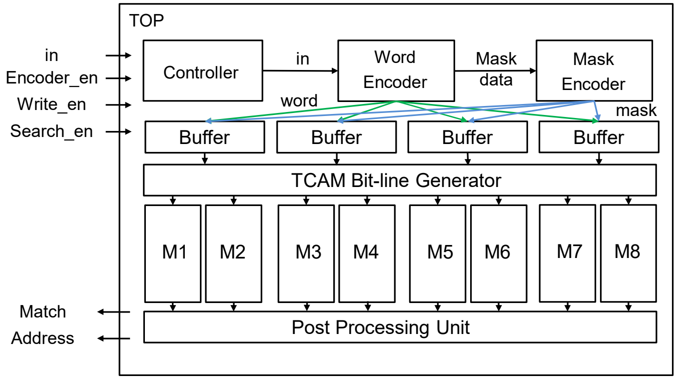
Similarity Search

- Which class does query belong to?
  - Iterative vs one-shot method



4

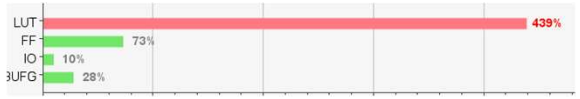
System Architecture



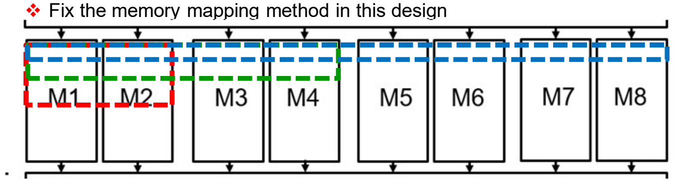
5

Direct Implementation

- Some Issue happened when implementing on FPGA directly
  - Too complicated
  - FPGA board has limited ALU resources



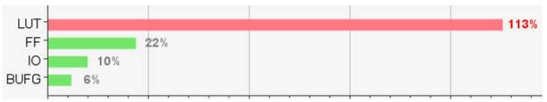
  - Reduce the use of MEM
  - Fix the memory mapping method in this design



6

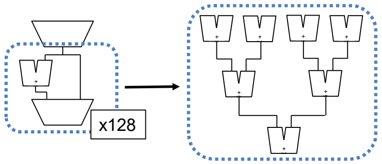
Direct Implementation

- Some Issue happened when implementing on FPGA directly
  - After removing unnecessary blocks
  - Still not simple enough



  - Rewrite memory macro

```
for (j = 1; j < 128; j=j+1) begin
  if(...)
    result[i][j]=1+result[i][j-1];
  else
    result[i][j]=result[i][j-1];
end
```



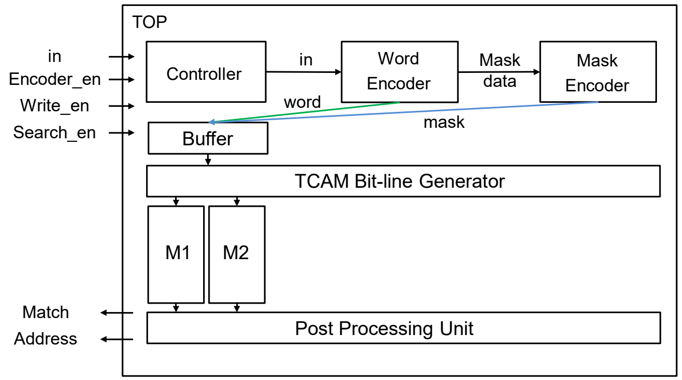
Large 3D array

Adder Tree

Smaller
Faster

7

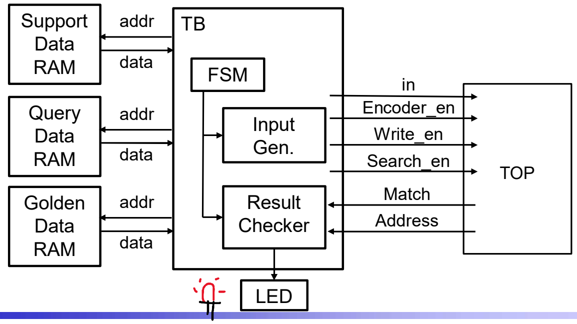
Revised System Architecture



8

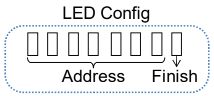
Input Stimuli Generation

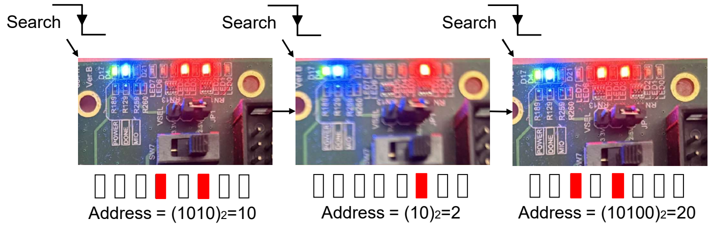
- Input stimuli were provided by testbench → Non-synthesizable
  - Rewrite testbench into a synthesizable module
  - Using RAM on FPGA board to preload data



Result

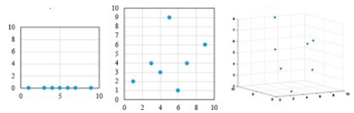
- After start signal is sent, TB receives MATCH signal from TOP once Search SW is triggered (active low)
  - LED shows the ADDRESS signal



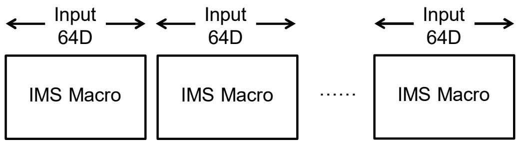


Curse of Dimensionality

- As dimensions increases, data become increasingly sparce, causing traditional algorithms to struggle

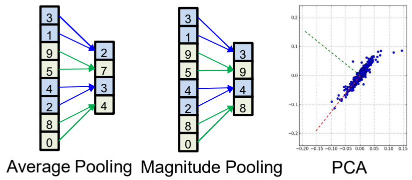


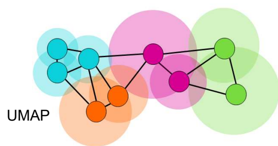
- The number of dimensions should meet the HW provided by MXIC
  - 64-dimensional INT3 L2-norm search

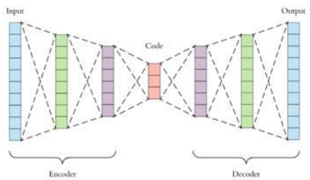


Methods of Dimension Reduction

- Linear
  - Pooling
    - Average Pooling
    - Magnitude Pooling
  - Principle Component Analysis
- Non-Linear
  - UMAP
  - Autoencoder (AE)



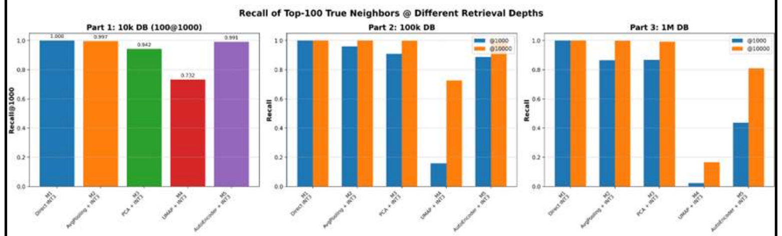




Closer points -> exponential decay to maintain local distance
Farther points -> inverse-polynomial to maintain global distance

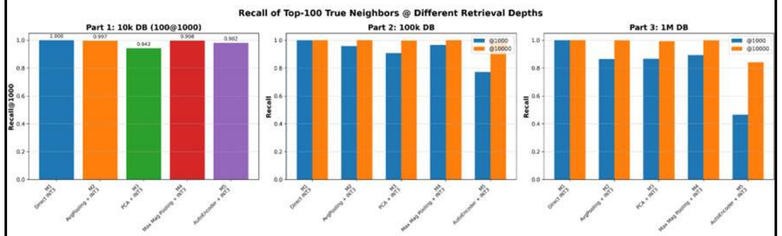
Experimental Results (SIFT-1M)

- Using SIFT-1M dataset
  - Vector dimension: 128 -> 64
  - Quantization method: min-max quantization
  - Avg. Pooling > PCA >= AE > UMAP



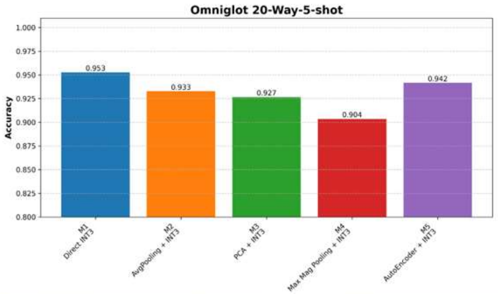
Experimental Results (SIFT-1M)

- Using SIFT-1M dataset
  - Vector dimension: 128 -> 64
  - Quantization method: min-max quantization
  - Since the performance of UMAP is worst, it is replaced by magnitude pooling
  - Avg. Pooling = Mag. Pooling >= PCA >= AE



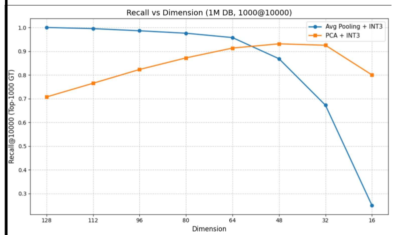
Experimental Results (Omniglot)

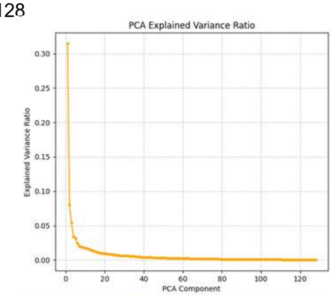
- Using Omniglot 20-way-5-shot experiment
  - Embedding dimension: 128 -> 64
  - Quantization method: min-max quantization
  - AE = Avg. Pooling = PCA >= Mag. Pooling



Analysis of PCA

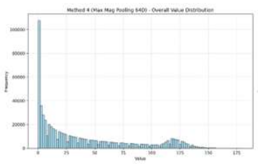
- PCA seems cannot win against average pooling
  - Is this true for all dimensions?
  - Why PCA achieves better recall rate when dim < 60
  - Too much noise from less representative dimensions
  - Low explained var. from dim 60 to 128



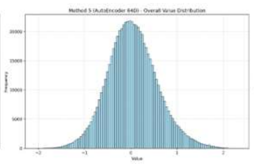


Data Distribution under Different Methods

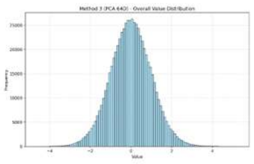
- What does the distributions look like after different dimension reduction methods
  - Some methods produces long-tail distributions
  - Others produces produces bell-shaped distributions
  - Can our quantization method fit all of these distributions?



Pooling



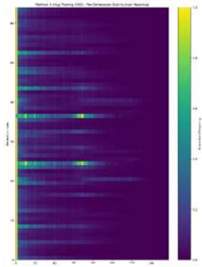
PCA



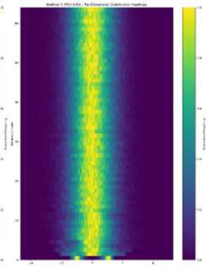
AutoEncoder

Data Distribution under Different Methods

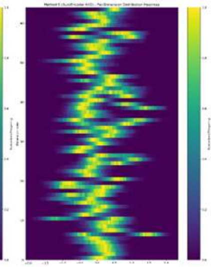
- What does the distributions look like after different dimension reduction methods
  - Some methods produces long-tail distributions
  - Others produces produces bell-shaped distributions
  - Can our quantization method fit all of these distributions?



Pooling



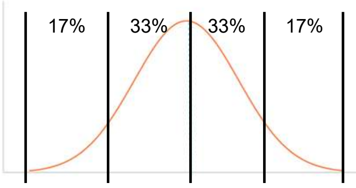
PCA



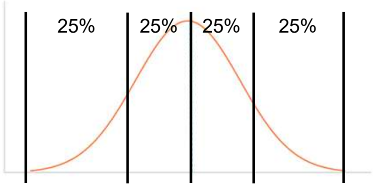
AutoEncoder

Quantization Makes Big Differences

- New quantization method : Percentile Quantization
  - Quantize data according to its percentile among all data
  - Non-linear quantization
  - Better to distinguish top-k nearest neighbors



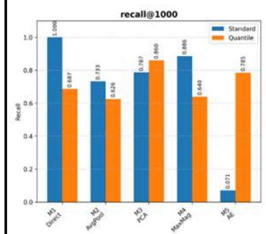
Min-max quantization

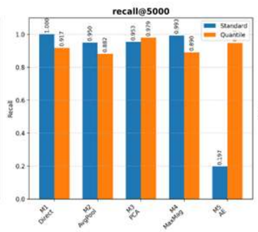


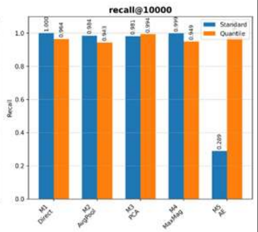
Percentile quantization

Experimental Results

- Percentile quantization boosts the recall rate of AutoEncoder method
  - Embedding dimension: 128 -> 64
  - **Blue: Min-max Quantization. Orange: Percentile Quantization**
  - Min-max -> better for long-tail distribution
  - Percentile -> better for bell-shaped distribution

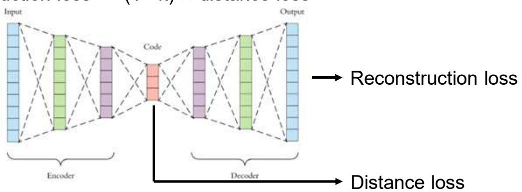






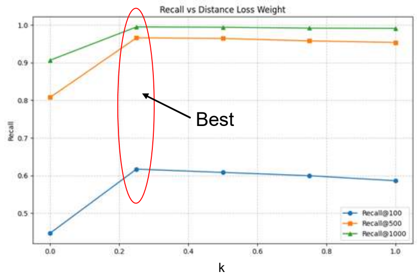
Modification of AutoEncoder

- Original AutoEncoders consider reconstruction loss as loss function
  - Construct a low dimension space
  - Can it preserve distance information?
- New loss function
  - Distance loss = MSE(distance before and after dimension reduction)
  - k × reconstruction loss + (1 - k) × distance loss



Experimental Results

- New loss function
  - k × reconstruction loss + (1 - k) × distance loss
  - Recall rate increased significantly by adding distance loss into loss function
  - Considering both reconstruction and distance info achieves best recall rate



Conclusion

- Embedding dimension reduction
  - Average pooling is simple but effective
- Analysis of PCA (Principle Component Analysis)
  - PCA outperforms average pooling when dimensions < 60
- Analysis of quantization method
  - Min-max quantization is better for pooling methods
  - Quantile is better for PCA and AutoEncoders
- Modification of AutoEncoder method
  - Preserving distance information increases recall rate
  - Best recall @ reconstruction : distance = 3 : 1