MULTIMEDIA COMPACT REPRESENTATIONS: APPROACHES, APPLICATIONS AND CHALLENGES

Jingkuan Song
jingkuan.song@gmail.com
Jingkuan Song is a full professor with University of Electronic Science and Technology of China (UESTC) from 2017. He was the winner of ACM SIGMM Rising Star Award 2021 for his continuous contributions to Multimedia Compact Representation and Analysis.

Xiaosu Zhu
xiaosu.zhu@outlook.com
Xiaosu Zhu is a Ph.D student at University of Electronic Science and Technology of China (UESTC) from 2020. His research interests include compact representation learning, multimedia retrieval and compression.

A. Introduction

Nowadays, a rapid increase on contents of images, videos and other media raises challenges on multimedia processing. Although modern computers or mobile devices have much more computational and storage resources than before, a system for handling billion-scale medias still requires excessive power beyond them. An efficient and performant processing pipeline is needed to meet such conditions.

Compact representations and corresponding technology are designed for such a large-scale multimedia processing, including image and video compression [1,2,3], retrieval [4,5,6], synthesis [7,8,9,27], etc. Other than full-precision data, compact representations use binary or low-bit values to formulate feature vectors, operators, and neural network weights. The storage is reduced by 10-100x while processing speed is accelerated via XOR, bit-count, lookup-table and other operators with hardware support [10]. By applying it to the above multimedia systems, the severe challenge of storage space and power consumption is largely alleviated.

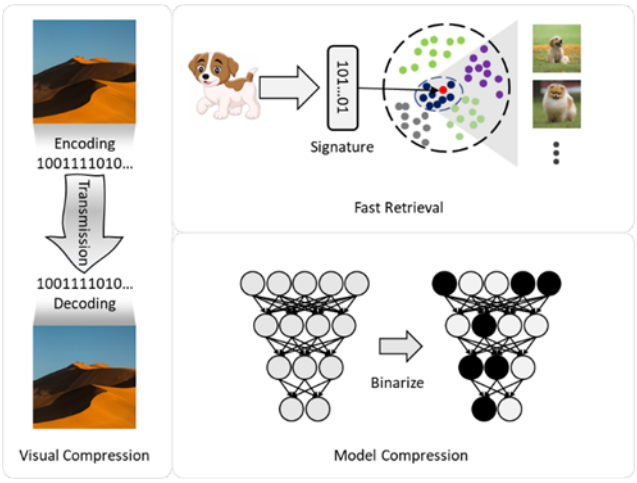

Figure 1. Typical applications of compact representations.

Today's compact representation algorithms, e.g., quantization and hashing are applied in a significant

wide space with heavy development. For visual compression, we could observe a steady improvement on compression ratio and quality from the classical JPEG image compression codec [1] to the latest neural image compression networks [2]. Similarly, a lot of vector libraries are emerged for large-scale data indexing and retrieval, e.g., faiss [6], milvus [11]. Thanks to the support of compact representations, a search in the billion-scale database is reduced to a few milliseconds and is conducted completely in memory. The recently popular tokenized image synthesis models VQ-GAN [8] and variants have initiated the new fashion of Artificial Intelligence Generated Contents (AIGC). The compact image tokens are obtained by a learnt vector quantizer, which hold abundant visual and semantic information to generate amazing arts, photographs and cartoons. Note that all the above techniques involve compact representations and make a great contribution to the current multimedia society in terms of power consumption, storage, etc. Next, a brief introduction of the fundamental techniques for them is brought.

B. Approaches

A regular vector held in computer is represented as an array of floating-point numbers, 32 bits for each. The goal of generating compact representation is to compress it into short binaries via hashing or quantization and keeps key information preserved.

B.1. Hashing

A hash function reduces high-dimensional data to a few fixed binary values, where similar data are allocated into similar hash values to keep the affinities. Therefore, the data distribution could be partially preserved by the converted binaries. It is commonly used for data clustering and nearest neighbor search. The famous Locality Sensitive Hashing (LSH) family [5] takes a concept to assign data into “buckets” while data collisions are maximized. Similarly, MinHash [12], Random Projection [13], etc. are designed in a data-independent way to achieve the goal, while Locality-Preserving Hashing performs in a data-dependent way. Furthermore, by utilizing semantic information to help data hashing, we could map data inputs according to their semantic similarity.

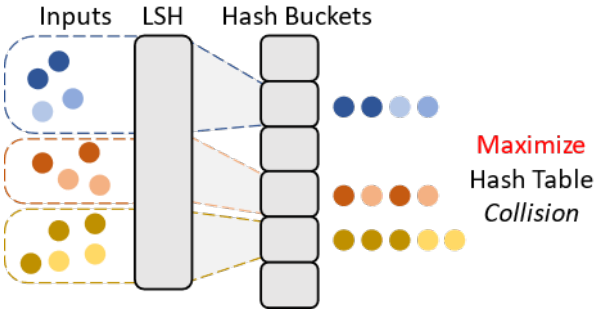

Figure 2. Demonstration of LSH family. They try to maximize collision of similar inputs in the hash table.

There is another approach to obtain hashing function that by several typical machine learning algorithms, called learning to hash [4]. Other than hand-crafted hashing rules, this kind of method uses mapping model, such as linear projection, to reduce feature dimension and binarize output by sign function. Neighborhood structure from input to output is preserved by defining objective function and performing optimization on mapping model. Then, the implicit hashing function is learnt and held in model.

Optimizing hashing models are not easy since finding an appropriate hashing output involves discrete and combinatorial optimization. Thus, a few heuristic algorithms are employed for it, e.g., coordinate descent in supervised discrete hashing, or a smooth relaxation over sign function to make it differentiable.

B.2. Quantization

Quantization acts like hashing since it also converts the raw full-precision data into a series of binary codes. While hashing directly makes comparison over hashing outputs, i.e., Hamming distance between different hashing codes, quantization uses a codebook to reconstruct original data while successive computations are happened in the reconstructed feature space.

Specifically, a scalar quantization is simple yet efficient since the implementation is rather naïve. It directly performs a rounding on floating-point numbers. Extended from this, a normal vector

quantization maps vectors to its nearest centroid, which is obtained by k-means [14]. Firstly, data are clustered by a few codewords, which are the representative vector on behalf of all neighborhoods. Combining all the codewords we obtain a codebook. Then, each input data is replaced by its nearest codeword and is stored by the index of codeword. Therefore, the quantization result of each vector is a single value, whose upper-bound is the total amount of codewords. It represents as fixed-length binaries in the storage.

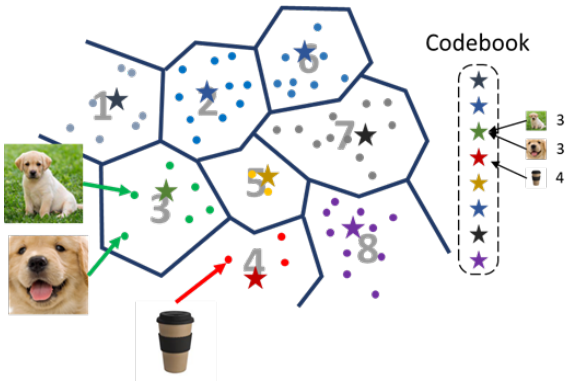

Figure 3. Demonstration of vector quantization. By clustering over input data, centers are collected as codebook. Every input is assigned to its nearest center.

The classical vector quantization is effective and has been used for a long time, but it has disadvantages in computational and space complexity, which is linearly related to the codebook size. The quantization precision is also limited with the total training data points since codebook size could not be larger than training set size. Therefore, product quantization is developed to fix the above issues by splitting raw data into a few orthogonal feature subspaces and quantizing separately by independent codebooks [15]. Therefore, the above complexity is reduced to logarithm-scale and the training becomes flexible. Further variants on vector quantization include composite quantization [16], additive quantization [17], etc. Note that the latter would be the super set of the former for the aforementioned algorithms with an improved performance.

C. Applications

As above demonstrates, we have two powerful tools for generating compact representations. Both of them have a long-time attraction and are under heavy development. Correspondingly, there are a lot of applications to integrate them into novel multimedia systems.

C.1. Visual Compression

Compact representations play the fundamental role in the pipeline of image compression, video compression and other widespread data compression techniques, since redundancies naturally exist in visual data and could be compressed by compact representations. For instance, a large area of blue sky contains low-frequency information mostly and could be represented by very few bits of a single color.

Conventional codecs for visual compression consist of transform coding, quantization, entropy coding, etc. Such practices have been confirmed as standards in modern image/video coding standards such as JPEG, H.264, VVC, etc [3].

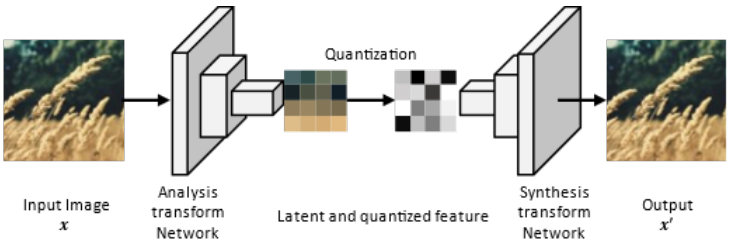

Figure 4. Typical operation diagram of a neural image compressor.

With the development of deep learning, compressing visual data by a neural network becomes a new fashion. It replaces the pre- and post-transform coding by the neural network to convert and reconstruct images to/from quantized latent features in a variational auto-encoder style. Meanwhile, a Straight-Through Estimator (STE) is applied on the quantization operation to enable end-to-end training via stochastic gradient descent. With the capacity of weights in neural networks and design of entropy models, deep image compression achieves superior performance than conventional ones. Recent works also extend quantization operations to multi-codebook vector quantization to significantly enhance coding efficiency [18]. With these methods, one could obtain 100-200x compression ratio compared to raw images without perceptible artifacts.

C.2. Fast Retrieval

Adopting hashing and quantization to perform fast retrieval is suitable since they could find approximate nearest neighbors in an efficient way with power of hardware acceleration. Moreover, the inverted file index could be built upon them to further increase search speed with a non-exhaustive search. Therefore, for million-scale and larger databases, several vector search software toolkits are developed for production, e.g., faiss, milvus. A huge number of applications are boosted by them, including but not limited to search engines, e-commerce, database indexing.

We would introduce a typical application: fast image retrieval, which digests images into short binary codes to formulate a database and provides ways to find similar images of a query. To achieve this, we could directly take image descriptors such as SIFT, GIST to extract features of images and obtain compact representations with the above methods. On the other hand, with the help of deep neural networks, an end-to-end fast image retrieval model could be made up with a network as backbone and a hashing/quantization layer on top of it [19]. Objective functions are designed to increase intra-class similarity and decrease inter-class similarity based on training image labels. Then, the images of same category are clustered together in order to organize the database. During retrieval, query image is firstly transformed into binary code in the same way and search by distance in the compact feature space, which could be accelerated by XOR (hashing) or look-up tables (quantization).

C.3. Tokenized Image Synthesis and Recognition

Another important application that involves compact representations is tokenized image synthesis. Formally, it utilizes and quantizes feature vectors extracted from a visual model, typically neural networks, and use the quantized results for further generation or recognition. Such an operation is dubbed tokenization.

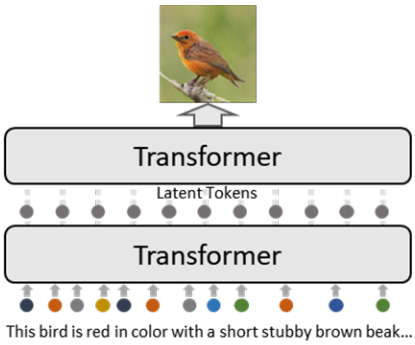

Figure 5. Demonstration of text to image synthesis by using tokens and transformers.

Specifically, VQ-VAE [7] constructs a VAE model while latent features produced by encoder are quantized by a codebook. By controlling the quantization value, the decoder could generate fake images. VQ-GAN uses transformer to enhance the generation ability upon the former. It also bridges multi-modal visual generation by unifying them into codewords.

With guide of quantized latent features of images, we would also perform visual recognitions. DALL-E [9], BeiT [20], CLIP [21], etc. bring the idea to produce classification results by formulating image tokens as sequence and performing sequence-to-sequence translation by transformers. They achieve satisfying performance and outperform convolutional networks of similar model size significantly.

C.4. Model Compression

The above applications utilize deep neural networks to obtain powerful and abundant latent representations and achieve better performance than conventional machine learning algorithms, thus become the current fashion. However, the model size and inference time of deep networks block them from running on devices with limited resources, such as mobile phones. Therefore, such demands make model compression become an important topic which benefits for deployment in real world.

Involved in model compression, binarizing or quantizing the model weights and intermediate activations are basic ways for reducing model size. It works since such values are also floating-point

vectors or matrices that could be converted into compact representations. By using hashing and scalar quantization techniques, the compression procedure is approximated by STE or relaxing for optimization. After training, a model with low-bit weights and activations would be obtained, which has a small size and the inference could be accelerated by XOR and bit-count operations [22]. To achieve good performance, knowledge distillation and weights regularization are adopted. The former forces small model to have similar activations with large models, while the latter prevents model to output trivial results [23].

D. Challenges

As a technique that receives attention in a long time, compact representations are now come with several powerful toolkits. Models or hand-crafted algorithms are well developed and could produce desired performance. However, there still has a few scenarios that previous works rarely study or could not handle well, especially for the realistic tasks. Next, we would explain them in detail.

Since discrete optimization over binaries is involved in compact representations, which is generally NP-hard, a lot of works focus on finding a solid solution to give near-optimal results. There are several remaining issues. For hand-crafted approaches, they either add constraints on it, which blocks them from the global-optima, or take a high time or space complexity for solving the problem. For deep learning based approaches, they would be trapped in local-optima where hashing layer produces trivial results or quantization falls into the “codebook collapse” problem. A few works try to tackle the above issues such as UNQ [24] and SQ-VAE [25], but a theoretical study on them remains rarely explored.

Due to the sensitivity of compact representations, i.e., data would be assigned with wrong binary value when inputs have distribution shift, current works result in bad performance especially when they meet novel inputs that are from other domains or categories. Note that a fine-tuning on models is not feasible since the new model would have gaps on representations with the old one, especially for tasks like fast retrieval, meaning that the retrieval database has to be re-built. Unfortunately, there are few works to study this problem [26].

This article makes a brief introduction to current advances in compact representations. Typical approaches such as hashing and quantization and corresponding optimization algorithms are explained. Applications that involve compact representations are introduced, while possible challenges are demonstrated. We could confirm that compact representations have wide use cases and potential improvements exist as the current research goes on.

References

[1] Gregory K. Wallace: The JPEG Still Picture Compression Standard. Commun. ACM 34(4): 30-44 (1991).
[2] Johannes Ballé, Valero Laparra, Eero P. Simoncelli: End-to-end Optimized Image Compression. ICLR 2017.
[3] Siwei Ma, Xinfeng Zhang, Chuanmin Jia, Zhenghui Zhao, Shiqi Wang, Shanshe Wang: Image and Video Compression With Neural Networks: A Review. IEEE Trans. Circuits Syst. Video Technol. 30(6): 1683-1698 (2020).
[4] Jingdong Wang, Ting Zhang, Jingkuan Song, Nicu Sebe, Heng Tao Shen: A Survey on Learning to Hash. IEEE Trans. Pattern Anal. Mach. Intell. 40(4): 769-790 (2018).
[5] Omid Jafari, Preeti Maurya, Parth Nagarkar, Khandker Mushfiqul Islam, Chidambaram Crushev: A Survey on Locality Sensitive Hashing Algorithms and their Applications. ArXiv preprint. 2102.08942 (2021).
[6] Jeff Johnson, Matthijs Douze, Hervé Jégou: Billion-Scale Similarity Search with GPUs. IEEE Trans. Big Data 7(3): 535-547 (2021).
[7] Aäron van den Oord, Oriol Vinyals, Koray Kavukcuoglu: Neural Discrete Representation Learning. NIPS 2017: 6306-6315.
[8] Patrick Esser, Robin Rombach, Björn Ommer: Taming Transformers for High-Resolution Image Synthesis. CVPR 2021: 12873-12883.
[9] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, Ilya Sutskever: Zero-Shot Text-to-Image Generation. ICML 2021: 8821-8831.
[10] H. Naseri and S. Timarchi, Low-power and fast full adder by exploring new XOR and XNOR gates. IEEE Trans. Very Large Scale Integr. Syst. 26(8): 1481-1493, 2018.
[11] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yinghao Zou, Jiquan Long, Yudong Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Ruiyi Jiang, Yi Wei, Charles Xie: Milvus: A Purpose-Built Vector Data Management System. SIGMOD Conference 2021: 2614-2627.
[12] Andrei Z. Broder, Moses Charikar, Alan M. Frieze, Michael Mitzenmacher: Min-Wise Independent Permutations (Extended Abstract). STOC 1998: 327-336.
[13] Ella Bingham, Heikki Mannila: Random projection in dimensionality reduction: applications to image and text data. KDD 2001: 245-250.
[14] Dana H. Ballard: An introduction to natural computation. Complex adaptive systems, MIT Press 2000, pp. I-XXII, 1-306.

[15] Hervé Jégou, Matthijs Douze, Cordelia Schmid: Product Quantization for Nearest Neighbor Search. IEEE Trans. Pattern Anal. Mach. Intell. 33(1): 117-128 (2011).
[16] Ting Zhang, Chao Du, Jingdong Wang: Composite Quantization for Approximate Nearest Neighbor Search. ICML 2014: 838-846.
[17] Artem Babenko, Victor S. Lempitsky: Additive Quantization for Extreme Vector Compression. CVPR 2014: 931-938.
[18] Xiaosu Zhu, Jingkuan Song, Lianli Gao, Feng Zheng, Heng Tao Shen: Unified Multivariate Gaussian Mixture for Efficient Neural Image Compression. CVPR 2022: 17591-17600.
[19] Tan Yu, Junsong Yuan, Chen Fang, Hailin Jin: Product Quantization Network for Fast Image Retrieval. ECCV 2018: 191-206.
[20] Hangbo Bao, Li Dong, Songhao Piao, Furu Wei: BEiT: BERT Pre-Training of Image Transformers. ICLR 2022.
[21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, Ilya Sutskever: Learning Transferable Visual Models From Natural Language Supervision. ICML 2021: 8748-8763.
[22] Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, Nicu Sebe: Binary neural networks: A survey. Pattern Recognit. 105: 107281 (2020).
[23] Haotong Qin, Ruihao Gong, Xianglong Liu, Mingzhu Shen, Ziran Wei, Fengwei Yu, Jingkuan Song: Forward and Backward Information Retention for Accurate Binary Neural Networks. CVPR 2020: 2247-2256.
[24] Stanislav Morozov, Artem Babenko: Unsupervised Neural Quantization for Compressed-Domain Similarity Search. ICCV 2019: 3036-3045.
[25] Yuhta Takida, Takashi Shibuya, Wei-Hsiang Liao, Chieh-Hsin Lai, Junki Ohmura, Toshimitsu Uesaka, Naoki Murata, Shusuke Takahashi, Toshiyuki Kumakura, Yuki Mitsufuji: SQ-VAE: Variational Bayes on Discrete Representation with Self-annealed Stochastic Quantization. ICML 2022: 20987-21012.
[26] Soumava Paul, Titir Dutta, Soma Biswas: Universal Cross-Domain Retrieval: Generalizing Across Classes and Domains. ICCV 2021: 12036-12044.
[27] Fuzhi Yang, Huan Yang, Jianlong Fu, Hongtao Lu, Baining Guo: Learning texture transformer network for image super-resolution. CVPR 2020: 5791-5800.