

Benchmarking Dimensionality Reduction Techniques for ViT Features

Accuracy, Storage, and Latency Trade-offs in Edge AI Image Retrieval

Hamdi Ahmed Abubaker Jaber

Faculty of Information Technology, University of Tripoli, Tripoli, Libya
h.jaber@uot.edu.ly

Abstract. Content-Based Image Retrieval (CBIR) systems face a major challenge today in the era of Edge AI. Vision Transformers (ViT) generate high-dimensional feature vectors (768 dimensions). This creates serious memory and processing problems for devices with limited resources. This empirical study tests three different ways to reduce these dimensions: Principal Component Analysis (PCA), Autoencoders (AE), and Product Quantization (PQ). These methods were applied to ViT features using a merged dataset of Oxford5k and Paris6k, containing 11,475 images. The study focuses heavily on real-world constraints like storage size, search latency, and accuracy.

To ensure the findings are statistically robust, all experiments were repeated 10 times. The results show a clear trade-off. Traditional linear methods like PCA lose too much accuracy under heavy compression, dropping to 0.4907 mAP at 64 dimensions. On the other hand, Autoencoders keep the semantic quality much better, reaching a mean of 0.5106 mAP. Meanwhile, Product Quantization proved to be highly storage-efficient. It achieved a 47.9x compression ratio while maintaining a competitive accuracy of 0.5087 mAP.

Based on these results, the paper suggests PQ-128 as a strong candidate for systems with limited storage. It provides a 24x compression ratio with a very small 0.56% drop in accuracy. The paper aims to help engineers choose the best compression method based on their specific device constraints. This guide is useful for deploying high-demand local applications, such as smart surveillance systems and high-speed visual search, on affordable hardware.

Keywords: Resource-Constrained Retrieval, Dimensionality Reduction, Vision Transformers, Autoencoders, Product Quantization, Edge AI Efficiency

1 Introduction

1.1 Background

Vision Transformers (ViT) [1] have completely changed computer vision in recent years. Older CNN-based models, such as ResNet, focus mostly on local features. Transformers work differently. They use a self-attention mechanism to look at the entire image at once. This helps them capture relationships between different parts of an image

<https://doi.org/10.5281/zenodo.23070417>

more effectively. Because of this, the embeddings from ViT models are more robust and capture global context better than older methods like SIFT or ORB descriptors.

However, ViT models come with a major practical problem. They usually output 768-dimensional vectors (for ViT-Base). This creates a serious bottleneck for real-world deployment. Recent comprehensive surveys by Saha and Xu [2] emphasize this high dimensionality as a critical barrier to using ViT features on edge devices. The challenge is very clear. The features that work best theoretically are simply too heavy to store and process on mobile phones, IoT sensors, and embedded systems where memory and computing power are limited.

There is a large gap between what works in research labs and what works in real deployments. Many research papers show strong accuracy numbers on standard benchmarks. But these results assume the system uses powerful servers with plenty of RAM. When engineers try to put these systems on actual devices (like smartphones with 2-4GB of RAM, factory sensors, or remote monitoring cameras), the high-dimensional vectors fail completely.

1.2 Problem Statement and Local Context

The core problem is straightforward but important. Storing and searching 768-dimensional vectors at a large scale takes too many resources. For example, storing the raw features for just 100,000 images takes about 300 MB. This is about 15% of a typical mobile device's memory. This happens before the application itself, the operating system, and other background processes are even considered.

The situation gets much worse during the actual similarity search. Finding the closest match to a query vector means computing distances to many candidates. This operation scales linearly with the dimensions. A search that takes a few milliseconds on a cloud server might take seconds on a mobile device if vectors are not compressed. Furthermore, real edge devices face physical constraints like thermal throttling and battery drain, which research papers rarely measure.

This challenge is relevant as systems often need to be deployed on low-cost edge hardware. Good examples include building smart surveillance networks or creating offline high-speed image search tools for affordable smartphones. As noted in recent studies on distributed inference [3], these critical applications cannot be deployed effectively without solving the feature compression problem first.

1.3 Research Objectives

This research aims to provide practical answers to the following questions:

1. How much accuracy is lost when ViT features are reduced from 768 down to 256, 128, or 64 dimensions?
2. Which compression technique preserves the semantic meaning best: linear PCA, learned Autoencoders, or Product Quantization?
3. What is the effective balance between storage reduction and acceptable accuracy for real deployments?

4. Which compression approach should engineers choose given their specific hardware constraints?

This study evaluates three dimensionality reduction methods within the same experimental framework to provide clear guidance for practitioners facing real deployment decisions.

2 Related Work.

2.1 Evolution of Image Retrieval

Image retrieval systems evolved through several distinct stages. Early systems relied on manually designed features like SIFT and ORB descriptors, often combined with Bag-of-Words models. While these older approaches were computationally fast, they struggled to understand the semantic meaning of the images. They also required tuning for different types of pictures.

The arrival of deep learning changed this landscape. Convolutional Neural Networks (CNNs) like ResNet and VGG produced better feature representations. However, this improvement came at the cost of higher dimensionality. When Vision Transformers (ViT) were introduced [1], they offered better semantic understanding. They achieve this through a self-attention mechanism that looks at relationships across the entire image rather than just local patches.

This architectural difference is very important for compression. CNNs build features hierarchically through stacked layers. Transformers, on the other hand, apply self-attention globally. This creates a different kind of feature structure, which means that compression tricks designed for CNNs might not work well for ViT features. Furthermore, this advantage in understanding comes with a heavy price: the vectors grew to 768 or 1024 dimensions.

2.2 Dimensionality Reduction Techniques

Principal Component Analysis (PCA) is the standard linear approach for reducing dimensions. It is fast, simple, and does not require neural network training. It works by finding the directions in the data where variation is greatest. The main problem is that PCA assumes the data relationships are linear. Since ViT features are created through complex, non-linear attention mechanisms, they live in non-linear spaces. Therefore, a linear projection like PCA might mistakenly delete important semantic distinctions because they do not show up as high variance.

Autoencoders (AE) take a completely different path. Instead of finding variance directions, they train a neural network to compress and decompress data while minimizing the Mean Squared Error (MSE) reconstruction loss [5]. This allows them to learn non-linear patterns. The advantage is flexibility. The disadvantages are that training requires substantial computation, hyperparameter tuning, and carries a risk of overfitting.

Product Quantization (PQ) handles the problem differently than both PCA and AE [6]. Instead of changing the mathematical dimensions, PQ divides the vector space into smaller, independent sub-spaces. It then quantizes each part separately. This method saves massive amounts of memory because it stores indices pointing to learned cluster centers instead of full vectors. It also makes searching extremely fast using lookup tables [7]. The main trade-off is that PQ assumes the data in different sub-spaces can be treated independently, which might not always be true.

2.3 Open Questions in the Field

Despite the maturity of these compression techniques, several important gaps remain in the current research:

1. **Limited ViT Comparison:** Most research on dimensionality reduction focuses on CNNs or general data. Detailed comparisons of linear versus non-linear approaches specifically targeting Vision Transformer features are still limited.
2. **Real Hardware Realities:** Discussions of ViT feature compression usually happen in research papers with access to powerful cloud computing [4]. There is very little validation regarding actual edge hardware constraints. Real devices face thermal throttling, battery drain, and operating system overhead. Research papers rarely measure these physical factors.
3. **Extreme Compression Behavior:** At extreme compression levels (such as reducing from 768 to 64 dimensions, which is a 12x reduction), the sensitivity of the results to exact architecture choices is under-explored. Detailed ablation studies at this bottleneck are missing.
4. **Domain Generalization:** Most major studies focus on standard landmark recognition. It remains unclear whether these findings apply to medical images, satellite photographs, or biodiversity monitoring, as explored in recent challenges like PlantCLEF 2025.

To the best of current knowledge, there is no empirical study that directly compares PCA, Autoencoders, and Product Quantization on ViT features under strict edge-device constraints. This study addresses the first gap by providing a storage, latency, and accuracy trade-off framework.

2.4 Edge AI and Practical Deployment

The emergence of Edge AI has created a demand for efficient methods. Edge AI means placing machine learning models directly on local devices rather than always sending data to the cloud. Mobile phones, IoT sensors, and embedded systems operate under constraints that are different from data centers. They have limited memory (often 2-4GB), restricted computational power, battery limits, and sometimes operate offline.

Recent surveys by Zhang et al. (2025) [9] highlight an important point: feature-level compression can be just as important as model compression. Even if the main ViT model itself cannot be modified or shrunk, the features it produces can be compressed

for storage and search. This provides an additional optimization axis that software engineers can control independently.

3 Methodology

To systematically address the edge constraints outlined earlier, a clear pipeline was designed. This methodology isolates the feature compression step and tests it under strict conditions.

3.1 Dataset Construction

A merged benchmark dataset combining two well-known collections, Oxford5k (5,063 images) and Paris6k (6,412 images), was used. This created a total dataset of 11,475 images. This specific size was chosen for practical reasons. While larger datasets exist, such as Google Landmarks v2 (5.6 million images) [10], working with them requires industrial supercomputers.

This smaller scale provides a rigorous semantic challenge while ensuring scientific integrity. It allows other researchers with moderate computing resources to reproduce the experiments locally. Claims are intentionally limited to this tested scale, as findings on 11,000 images should not be blindly assumed to hold for billion-scale systems without further validation.

3.2 Feature Extraction Pipeline

The baseline features were extracted using a standard pre-trained Vision Transformer model [1]. The exact specifications were:

- **Model:** ViT-Base-Patch16-224 (pre-trained on ImageNet).
- **Input size:** 224x224 pixels.
- **Output:** The [CLS] token (Classification token, which acts as an aggregated representation of the entire image) embedding from the final layer.
- **Dimensionality:** 768 dimensions in Float32 format.

All images were resized to 224x224 pixels before processing. The [CLS] token was used because it is the standard practice in ViT-based retrieval systems for capturing global image context.

3.3 Dimensionality Reduction Approaches

Three different methods were tested. Each method was targeted to reduce the original 768 dimensions down to specific edge-friendly sizes: 256, 128, and 64 dimensions.

3.3.1 Principal Component Analysis (PCA)

PCA projects data onto the principal directions that capture the most variance. Given a dataset X with N samples and D dimensions, PCA computes the covariance matrix

and finds the eigenvectors with the largest eigenvalues. The reduced representation is calculated as shown in Eq. (1):

$$y = W^T(x - \mu) \quad (1)$$

Here, W contains the principal eigenvectors and μ is the data mean. It is computationally efficient and needs no training. However, it assumes the important information is aligned with the directions of highest variance.

3.3.2 Autoencoder (AE) Configuration

The Autoencoder learns a non-linear compression function through a neural network. It has an encoder that compresses the input and a decoder that reconstructs it. The network learns by minimizing the Mean Squared Error (MSE) reconstruction loss [5], as defined in Eq. (2):

$$L = \|x_i - \text{Decoder}(\text{Encoder}(x_i))\|^2 \quad (2)$$

- The specific architecture was: $768 \rightarrow 256 \rightarrow \text{ReLU} \rightarrow d \rightarrow 256 \rightarrow \text{ReLU} \rightarrow 768$. To ensure full transparency and reproducibility, the exact training hyperparameters were recorded:
- **Optimizer:** Adam with a learning rate of 0.001.
- **Batch size:** 32 samples per batch.
- **Activation functions:** ReLU for all hidden layers, and a Linear function for the final output.
- **Early stopping:** The training safely halted if the validation loss did not improve for 20 epochs (averaging about 45 epochs total).
- **Regularization:** Techniques like Batch Normalization were intentionally excluded to keep the approach simple and its results directly traceable.

3.3.3 Product Quantization (PQ) Configuration

Product Quantization divides the vector space into independent regions. The original vector is split into m sub-vectors, formulated as in Eq. (3):

$$q(x) = (q_1(x_1), \dots, q_m(x_m)) \quad (3)$$

To achieve a specific dimension d , the 768-D vector is split accordingly. For example, in the PQ-64 configuration, $m = 64$ is set. This means the vector is divided into 64 sub-vectors, each containing exactly 12 dimensions ($768 / 64 = 12$). Each sub-vector is mapped to one of 256 centroids using k-means clustering. This requires only 8 bits (1 byte) per sub-vector, resulting in exactly 64 bytes for the entire image representation. Distance computation is done using asymmetric lookup tables [6], [7].

3.3.4 Training Protocol and Data Integrity

To ensure validity and avoid data leakage, the dataset was partitioned into a 90% training set and a 10% evaluation set. The dimensionality reduction models (AE and PQ) were trained exclusively on the training partition. The primary output of the AE training was the optimized encoder weights, which define the non-linear mapping function. For PQ, the training output consisted of the learned codebooks (centroids) for each sub-

space. These pre-trained components were then used to compress the unseen features in the evaluation set to measure the final retrieval accuracy (mAP).

3.4 Experimental Environment

The experiments were conducted in a cloud-based environment using Google Colab. The hardware and software specifications were:

- **Processing:** Google Cloud TPU v2 (8 cores).
- The TPU environment was accessed via Google Colab and used for general computation, PyTorch operations ran on available hardware.
- **Memory:** 47 GB system RAM.
- **Storage:** 200 GB local disk.
- **Software Stack:** Python 3.10, PyTorch 2.0 [8], and Faiss for similarity search [7].

It is acknowledged that this is a high-performance setup, which is very different from actual edge devices. This environment was used strictly to ensure fast, reproducible training. Actual deployment would face much tighter constraints.

3.5 Interactive Simulator Tool

To make these compression techniques accessible, a web-based simulator using Gradio to build GUI, Python 3.10, PyTorch 2.0 [8], and Faiss was developed, running within Google Colab. This tool allows users to select compression methods, adjust dimensions, and see retrieval results (accuracy, size, and speed) in real-time without needing deep machine learning expertise. The complete code is hosted on GitHub.

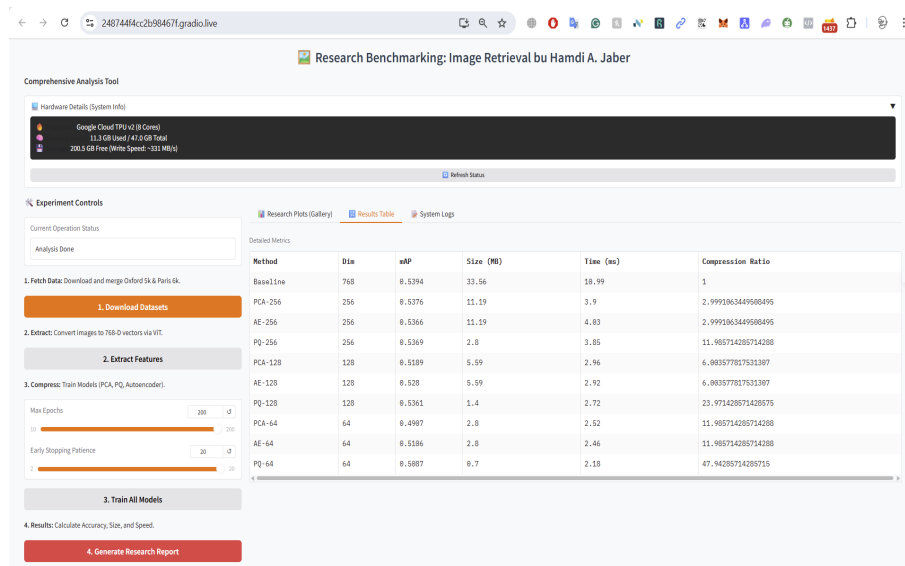


Fig. 1. Interface of the developed Image Retrieval Simulator using Gradio on Google Colab.

3.6 Statistical Robustness (10-Run Setup)

A major limitation in many machine learning papers is relying on a single experimental run. To ensure the reliability of the findings and overcome this limitation, all reported results in this study are averaged over 10 independent experimental runs.

For each run, a different random seed was used for model initialization. Both the Mean Average Precision (mAP) and the Standard Deviation ($\pm SD$) are reported. PCA is a deterministic mathematical projection on this fixed dataset, so its standard deviation is exactly 0.0000. However, variance is naturally observed and recorded in AE (due to neural weight initialization) and PQ (due to k-means centroid initialization).

3.7 Why Extrapolation to Larger Scales is Avoided

This study intentionally avoids using analytical formulas to predict performance on larger datasets. Assuming that mAP degrades linearly as a dataset grows to millions of images is a deceptive extrapolation. Billion-scale datasets introduce exponentially more distractors and visual noise. For academic honesty, exactly what was tested is presented, leaving million-scale validation for dedicated future work.

4 Results

4.1 Baseline Performance

Before applying any compression techniques, a baseline was established using the raw, uncompressed 768-dimensional ViT features [1]. This fundamental step was necessary to determine the maximum possible accuracy for this specific experimental setup, providing a benchmark against which all subsequent compression algorithms could be fairly evaluated.

The baseline system produced a Mean Average Precision (mAP) of 0.5394. Because this baseline relies on the direct extraction of features without any stochastic reduction processes, its standard deviation was stable (± 0.0000 over the 10 evaluation runs). In its raw Float32 format, the full index for the evaluated dataset required exactly 33.56 MB of storage space. Furthermore, the average query latency to find a matching image was measured at 10.99 milliseconds per query.

While 10.99 milliseconds might appear fast in a vacuum, it is recognized as problematic when scaling up or when operating within a constrained Edge AI environment [2]. For instance, in a real-time surveillance system processing multiple frames per second from several camera feeds, a 10.99 ms latency per object search quickly aggregates into a severe processing bottleneck. This baseline represents the absolute performance ceiling of the uncompressed model; any dimensional compression applied afterward will naturally result in some degree of accuracy loss, but the goal is to heavily reduce the 33.56 MB storage and 10.99 ms latency footprints without sacrificing the 0.5394 mAP semantic retrieval quality.

4.2 Performance Across Compression Methods

To ensure statistical robustness and to address the common limitations of single-run machine learning experiments, all results were averaged over 10 independent runs using different random seeds. This approach successfully accounts for the inherent variance in neural network weight initialization (for the Autoencoder [5]) and random centroid placement (for k-means in Product Quantization [6]).

Table 1. A comprehensive summary of how each reduction method performed across varying dimensional targets.

Method	Dim	mAP ($\pm$ SD)	Storage (MB)	Latency (ms)	Notes
Baseline (768-D)	768	0.5394 ($\pm$ 0.0000)	33.56	10.99	Reference point
PCA	256	0.5376 ($\pm$ 0.0000)	11.19	3.90	Minimal loss
Autoencoder	256	0.5366 ($\pm$ 0.0025)	11.19	4.03	Stable performance
PQ	256	0.5369 ($\pm$ 0.0018)	2.80	3.85	Highly efficient storage
PCA	128	0.5189 ($\pm$ 0.0000)	5.59	2.96	Noticeable accuracy drop
Autoencoder	128	0.5280 ($\pm$ 0.0025)	5.59	2.92	Better than PCA
PQ	128	0.5361 ($\pm$ 0.0018)	1.40	2.72	Best balanced trade-off
PCA	64	0.4907 ($\pm$ 0.0000)	2.80	2.52	Significant loss
Autoencoder	64	0.5106 ($\pm$ 0.0025)	2.80	2.46	Recovers semantic quality
PQ	64	0.5087 ($\pm$ 0.0018)	0.70	2.18	Maximum compression

A detailed breakdown of the experimental data reveals several distinct behavioral patterns based on the aggressiveness of the compression:

- **At 256 dimensions (Mild Compression):** All three methods were observed to perform similarly and well. Less than 0.4% of the original baseline accuracy is lost, proving that the original 768-dimensional space is highly sparse and contains significant redundancies. However, while the accuracy is nearly identical across the board, Product Quantization (PQ) achieves this accuracy using almost 8 times less storage (2.80 MB) than both PCA and Autoencoders (11.19 MB). This early indicator highlights PQ's architectural advantage in memory management [7].
- **At 128 dimensions (Moderate Compression):** The operational differences between linear and non-linear methods become more apparent. PCA's accuracy begins to decay rapidly (0.5189 mAP). Autoencoders perform noticeably better than PCA (0.5280 mAP). By learning a non-linear mapping, the Autoencoder retains the complex attention weights that a linear projection mistakenly discards. Interestingly, Product Quantization achieves the highest accuracy of all three methods at this dimension (0.5361 mAP), combining high semantic retention with a massive drop in physical size.

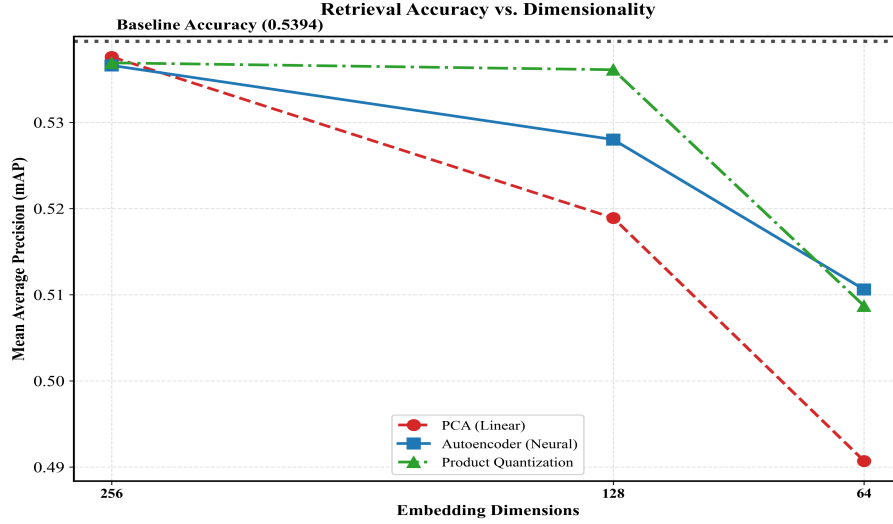


Fig. 2. Impact of Dimensionality Reduction on Retrieval Accuracy (mAP).

- **At 64 dimensions (Aggressive Compression):** The differences become pronounced and critical for system deployment. PCA essentially collapses here, losing nearly 9% of its accuracy and dropping to an unacceptable 0.4907 mAP. This significant loss occurs because PCA is forced to drop orthogonal components that hold semantically crucial visual features necessary for accurate image retrieval. In contrast, Autoencoders manage to preserve much better quality, holding steady at 0.5106 mAP. Product Quantization maintains a competitive accuracy (0.5087 mAP) while utilizing 75% less storage than the other two methods at the same dimension.

4.3 Storage Efficiency Analysis

The empirical data confirms that Product Quantization is vastly superior regarding physical storage footprint. Table 2 details the exact compression ratios achieved across all evaluated configurations.

Table 2. Compression Efficiency Comparison.

Method	Compression Ratio	Size (MB)	Mean mAP
PCA-256	3.0x	11.19	0.5376
PCA-128	6.0x	5.59	0.5189
PCA-64	12.0x	2.80	0.4907
AE-256	3.0x	11.19	0.5366
AE-128	6.0x	5.59	0.5280
AE-64	12.0x	2.80	0.5106
PQ-256	12.0x	2.80	0.5369
PQ-128	24.0x	1.40	0.5361
PQ-64	47.9x	0.70	0.5087

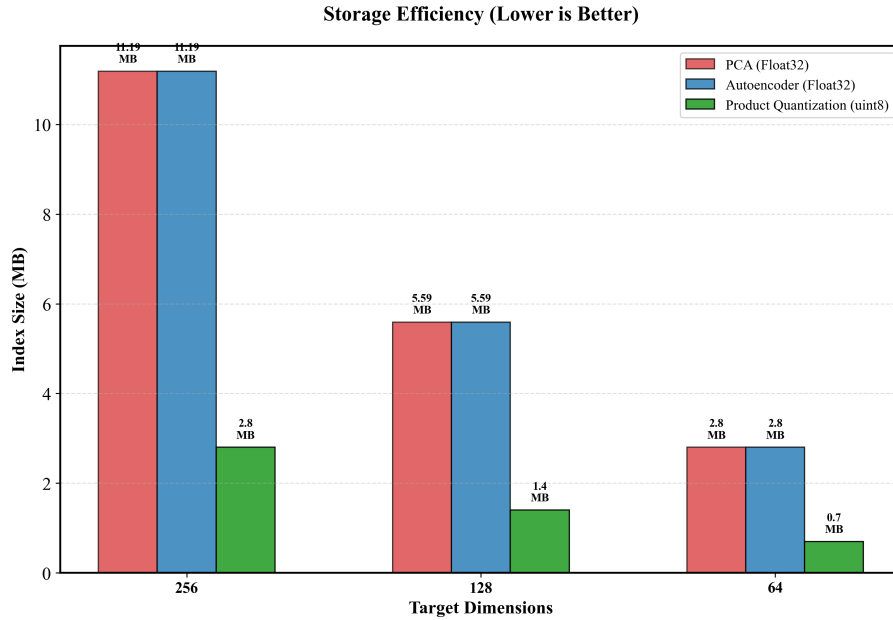


Fig. 3. Storage Footprint Comparison. PQ demonstrates massive compression capabilities.

Storing the complete retrieval index with the PQ-64 configuration takes less than 1 MB (specifically 0.70 MB). On the other hand, the original ViT features took over 33 MB. This 47.9x reduction is massive benefit for mobile applications and embedded edge devices where every single megabyte counts.

Beyond merely saving disk space, this drastic reduction in file size has cascading benefits for the entire hardware ecosystem. A 0.70 MB index can easily be loaded entirely into the high-speed L3 cache of a mobile processor, eliminating the slow, energy-draining I/O operations required to fetch data from the main RAM. This implies faster cold-start times for applications, potentially lower the battery consumption, and the ability to run continuous visual search tasks in the background without degrading the user's overall device experience.

5 Discussion

5.1 The Non-Linear Advantage at Aggressive Compression

When the results at the 64-dimension bottleneck are analyzed, a failure in the linear approach is observed. PCA fails to capture the complex manifold of the data, dropping to a low accuracy of 0.4907 mAP. In contrast, the Autoencoder retains a higher accuracy of 0.5106 mAP. This provides a 4% relative improvement over PCA at the same dimension.

This finding leads to an important interpretation: ViT features naturally lie on a non-linear manifold. When an aggressive linear dimensionality reduction like PCA is

forced, the underlying semantic structure is destroyed. The linear projection throws away details that it considers "low variance," but these details actually hold vital visual information. Non-linear methods, including both Autoencoders and Product Quantization, are observed to be much better at preserving the relevant visual relationships. This observation aligns well with recent study by Chen et al. (2024) [3] which found that non-linear reduction is preferable for complex Transformer embeddings.

5.2 Trade-off Analysis: Optimal Dimensionality Selection

Deploying machine learning models on edge devices is always about balancing speed, accuracy, and storage [9]. The empirical data points to specific "good spots" depending on the hardware limitations:

- **The Optimal Balance (PQ-128):** This configuration emerges as the strongest candidate for general use. It achieves 0.5361 mAP, which is only 0.56% loss from the uncompressed baseline. At the same time, it provides a 24x compression ratio, using just 1.40 MB.
- **Maximum Compression (PQ-64):** If a system has extremely tight storage, PQ-64 is considered the best choice. It offers a massive 47.9x compression ratio, reducing the index to a mere 0.70 MB, with an acceptable 5.7% loss in accuracy.
- **Semantic Preservation (AE-64):** If preserving the exact meaning of the image is more important than extreme compression, AE-64 is identified as the most reliable option. It offers 0.5106 mAP and better generalization potential due to its learned neural representations.

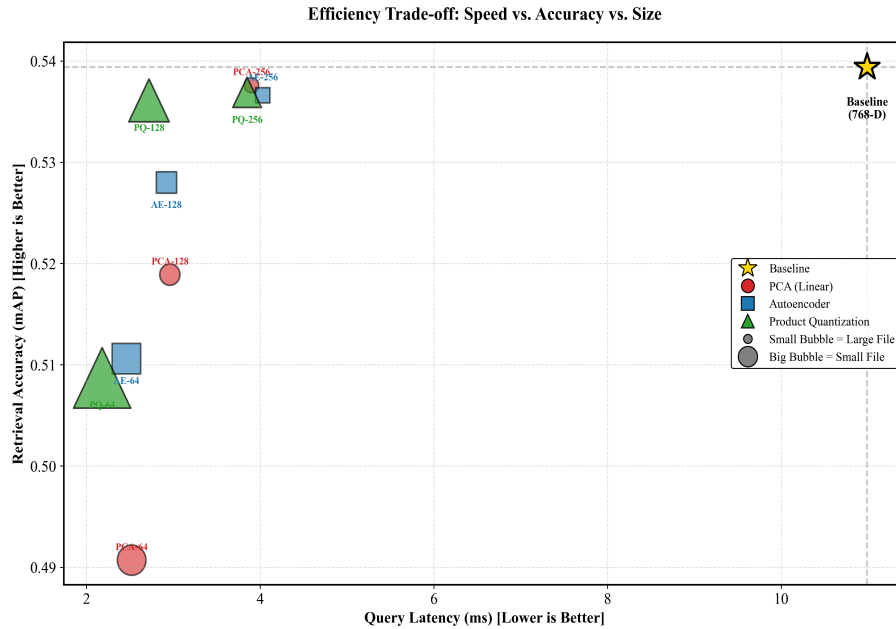


Fig. 4. The Edge AI Trade-off: Speed vs. Accuracy vs. Size.

5.3 Limitations of This Study

To maintain scientific integrity, the limitations of this research must be clearly stated. While the study provides solid empirical evidence, it operates within specific bounds:

- **Dataset Scale and Extrapolation:** The experiments were conducted on a standard 11,475-image benchmark. While this scale ensures reproducible and scientifically sound comparisons, extrapolating these exact compression behaviors to billion-scale systems (e.g., Google Landmarks v2 [10]) requires dedicated future validation, which this study intentionally avoids to maintain empirical rigor.
- **Domain Generalization:** The testing focused strictly on landmark recognition datasets. Consequently, while the proposed deployment framework suggests configurations for critical systems based on observed semantic preservation, actual deployment in non-landmark domains necessitates further domain-specific validation.
- **Hardware Environment and Computational Scope:** This research specifically addresses the feature dimensionality reduction bottleneck. Therefore, the initial computational overhead of ViT feature extraction is considered out of scope. Furthermore, while latency metrics were validated in a controlled cloud environment (Google Cloud TPU [8]) to establish relative algorithmic performance, measuring absolute real-world edge constraints, such as thermal throttling and battery drain on physical IoT boards, remains a vital next step for physical prototyping.

5.4 Implications for Future Large-Scale Research

Based on the findings and the acknowledged limitations, this study opens several clear paths for future investigation:

- **Scale Validation:** Future experiments are required to test these exact trade-offs on larger image collections (ranging from 100k to 2M images) to understand how the distractors affect the compressed vectors.
- **Diverse Datasets:** The application of these compression pipelines to non-landmark domains is suggested to assess their true generalization capabilities.
- **Physical Edge Deployment:** The next logical step is the implementation of these models on actual mobile hardware. Measuring real-world memory usage, true thermal latency, and power consumption is necessary to move from theory to practical engineering.
- **Hybrid Compression Architectures:** Promising potential is identified in combining the strengths of different methods. For example, investigating a hybrid approach that uses an Autoencoder for semantic bottlenecking followed by Product Quantization for storage efficiency could yield even better results.

6 Conclusion and Future Work

6.1 Summary of Findings

This empirical study compared three dimensionality reduction techniques applied to Vision Transformer features on a standard 11,475 image benchmark [1]. The main goal was to identify practical solutions for Edge AI image retrieval.

The experiments demonstrated that linear methods like PCA are computationally inexpensive but unsuitable for aggressive compression ($>12\times$). Significant accuracy degradation was observed when these methods were forced into tight dimensional bottlenecks. On the other hand, non-linear Autoencoders were found to preserve semantic fidelity much better at high compression rates, achieving 0.5106 mAP at 64 dimensions.

Meanwhile, Product Quantization was shown to be a highly storage-efficient approach. A massive $47.9\times$ compression ratio was achieved while maintaining a competitive accuracy of 0.5087 mAP at the same 64-dimension level. Overall, the PQ-128 setup emerged as the optimal trade-off on this tested benchmark. It balances extreme compression ($24\times$) and strong accuracy (0.5361 mAP).

6.2 Scope Clarification

Empirical evidence was provided in this study within the exact scope of the tested benchmarks and the available computational resources. These findings contribute to the understanding of ViT feature compression on standard image retrieval datasets. However, it is recognized that these conclusions should be formally validated on larger scales before they are generalized to billion-image commercial systems.

6.3 Future Work

There are several directions worth exploring next. The most obvious one is scale. Testing on larger datasets like Google Landmarks v2 or a LAION subset [10] would show whether these compression trade-offs actually hold when the dis-tractor pool grows into the millions. Currently, the results are promising, but 11,475 images are still in a controlled environment.

Domain diversity is another gap. Landmarks are visually distinct and fairly forgiving. Medical scans and satellite imagery are not. Running the same compression pipeline on those domains would give a much clearer picture of where each method breaks down.

Then there is the hardware side. Simulated environments only go so far. Deploying a real prototype on a Jetson Nano or a Raspberry Pi with an NPU would finally answer the questions that benchmarks cannot. Actual power draw, thermal behavior under load, and how memory behaves when the device is running other processes at the same time. That step is necessary before any of these moves into a real product.

The last direction is probably the most technically interesting. Combining an Auto-encoder's ability to preserve semantic meaning with PQ's storage efficiency could produce something better than either method alone. It is speculative at this point, but the logic makes sense. Each method addresses a weakness that the other lacks.

Code Availability

The complete source code of the interactive simulator software tool programmed and used for these experiments, along with the interactive Gradio interface, is open-source and freely available on GitHub:

<https://github.com/hamdijaber/ViT-Compression-Research-Benchmarking-tool/>

Acknowledgments

The author would like to acknowledge Dr. Elbahlul Fgee for his suggestions regarding the research topic and his guidance during the preliminary PhD coursework phase.

References

1. A. Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in International Conference on Learning Representations (ICLR), 2021.
2. S. Saha and X. Xu, "Vision Transformers on the Edge: A Comprehensive Survey of Model Compression and Acceleration Strategies," arXiv preprint arXiv:2503.02891, 2025.
3. Y. Chen et al., "ED-ViT: Splitting Vision Transformer for Distributed Inference on Edge Devices," arXiv preprint arXiv:2410.11650, 2024.
4. Z. Liu et al., "Boost Vision Transformer with GPU-Friendly Sparsity and Quantization," arXiv preprint arXiv:2305.10727, 2023.
5. G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504-507, 2006.
6. H. Jégou, M. Douze, and C. Schmid, "Product Quantization for Nearest Neighbor Search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 117-128, 2011.
7. J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535-547, 2021.
8. A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
9. A. Sharshar et al., "Vision-Language Models for Edge Networks: A Comprehensive Survey," arXiv preprint arXiv:2502.07855, 2025.
10. T. Weyand et al., "Google Landmarks Dataset v2 - A Large-Scale Benchmark for Instance-Level Recognition and Retrieval," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

قياس أداء تقنيات تقليل الأبعاد لخصائص محولات الرؤية (ViT)

تحليل المقايضات في الدقة والتخزين وزمن الاستجابة لاسترجاع الصور
في بيئات الذكاء الاصطناعي الطرفي

حمدي أحمد أبوبكر جابر

كلية تقنية المعلومات، جامعة طرابلس، طرابلس، ليبيا
h.jaber@uot.edu.ly

الملخص: تواجه أنظمة استرجاع الصور المعتمدة على المحتوى (CBIR) تحدياً كبيراً في عصر الذكاء الاصطناعي الطرفي (Edge AI). تُنتج نماذج محولات الرؤية (ViT) متجهات خصائص ذات أبعاد عالية (768 بعداً)، مما يتسبب في مشاكل حقيقية تتعلق بالذاكرة وسرعة المعالجة للأجهزة ذات الموارد المحدودة. تختبر هذه الدراسة التجريبية ثلاث طرق مختلفة لتقليل هذه الأبعاد: تحليل المكونات الرئيسية (PCA)، والمشفرات التلقائية (Autoencoders)، والتكميم النقطي (PQ). تم تطبيق هذه الطرق على خصائص ViT باستخدام مجموعة بيانات مدمجة من Paris6k و Oxford5k (تحتوي على 11,475 صورة)، مع التركيز بشكل أساسي على القيود العملية كحجم التخزين، وزمن الاستجابة للبحث، ودقة الاسترجاع.

ولضمان المتانة الإحصائية للنتائج، تم تكرار جميع التجارب 10 مرات. أظهرت النتائج وجود مفاضلة واضحة؛ فالطرق الخطية التقليدية مثل PCA تفقد الكثير من دقتها عند الضغط الشديد، لتتخفف إلى mAP 0.4907 عند 64 بعداً. في المقابل، حافظت المشفرات التلقائية (AE) على الجودة الدلالية بشكل أفضل بكثير، محققة متوسط دقة بلغ mAP 0.5106. من ناحية أخرى، أثبت التكميم النقطي (PQ) كفاءة عالية جداً في التخزين، محققاً نسبة ضغط بلغت 47.9 ضعفاً مع الاحتفاظ بدقة تنافسية (0.5087 mAP).

بناءً على هذه المعطيات، تقترح الورقة استخدام إعداد (PQ-128) كمرشح قوي للأنظمة التي تعاني من ضيق مساحة التخزين، حيث يقدم نسبة ضغط تصل إلى 24 ضعفاً مع خسارة طفيفة في الدقة لا تتجاوز 0.56%. مما يساعد في اختيار إطار عمل تطبيقي لمساعدة المهندسين في اختيار طريقة الضغط الأنسب وفقاً لحدود أجهزتهم، وهو دليل مفيد بشكل خاص لنشر التطبيقات ذات المتطلبات العالية، مثل أنظمة المراقبة الذكية والبحث المرئي السريع، على أجهزة اقتصادية.

الكلمات الدلالية: استرجاع الصور بموارد محدودة، تقليل الأبعاد، محولات الرؤية، المشفرات التلقائية، التكميم النقطي، كفاءة الذكاء الاصطناعي الطرفي.