Efficiency Analysis of Dimensionality Reduction for ViT Features in Resource-Constrained Image Retrieval

LIBYAN ACADEMY FOR POSTGRADUATE STUDIES

School of Applied Science and Engineering

Fall 2025

PhD Student: Hamdi A. Jaber (ID: 250100836)

Supervisor: Dr. Elbahlul Fgee

December 2025

ABSTRACT

Content-Based Image Retrieval (CBIR) systems struggle with a significant challenge in the modern era of Edge AI: Vision Transformers (ViT) generate high-dimensional feature vectors of 768 dimensions, which create serious problems for resource-constrained devices. This empirical study examines how three different approaches to reducing dimensions—Principal Component Analysis (PCA), Autoencoders (AE), and Product Quantization (PQ)—work when applied to ViT features. The study utilizes a merged dataset of Oxford5k and Paris6k, containing 11,475 images, and focuses on real-world constraints like storage, speed, and accuracy.

The results show that traditional linear methods like PCA lose too much accuracy when compressed aggressively, dropping to 0.4907 mAP at 64 dimensions. In contrast, Autoencoders maintain better semantic quality, reaching 0.5106 mAP, which is closer to the operational requirements of semantic retrieval. Product Quantization proves to be the most storage-efficient option, achieving a 47.9$\times$ compression ratio while keeping competitive accuracy at 0.5087 mAP.

Based on these findings, the paper suggests that for systems with limited storage, PQ-128 offers the best balance—24$\times$ compression with minimal accuracy loss of 0.56%. For critical applications requiring better accuracy preservation, Autoencoders at 64 dimensions are preferable. These results provide a practical guide for engineers deciding which compression method to use depending on their specific device constraints.

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

1 INTRODUCTION

1.1 Background

Vision Transformers have changed how computer vision works over the past few years. Unlike older CNN-based approaches such as ResNet [6] that focus on local features, Transformers use a self-attention mechanism to look at the entire image at once [19]. This allows them to capture relationships between different parts of an image more effectively. The embeddings that ViT models produce are therefore more robust and capture global context better than traditional methods like SIFT or ORB descriptors.

However, ViT models come with a significant practical problem. These models typically output vectors of 768 dimensions (for ViT-Base), which creates a major bottleneck for deployment. Recent comprehensive surveys by Saha and Xu [16] emphasize this as one of the critical barriers to utilizing ViT features on edge devices. The challenge is clear: the features that work best theoretically are expensive to store and process on mobile phones, IoT devices, and embedded systems where memory and computation are limited.

The gap between what works in research labs and what works in real-world deployments is significant. Most research papers show strong accuracy numbers on standard benchmarks, but these results assume access to powerful servers with plenty of memory and processing power. When engineers try to move these systems to actual devices—smartphones with 2-4GB of RAM, sensors on factory equipment, or monitoring systems in remote locations—the high-dimensional vectors become impractical.

1.2 Problem Statement

The core problem is straightforward but important: storing and searching 768-dimensional vectors at any meaningful scale requires excessive resources. For example, storing features for just 100,000 images would consume 300 MB in raw form—roughly 15% of a typical mobile device's total memory. This is before considering the application itself, the operating system, and other necessary background processes.

The situation gets worse when considering actual similarity search. Finding the closest match to a query vector requires computing distances to many candidates, and this operation scales with the number of dimensions. A search that might take a few milliseconds on a server could take seconds on a mobile device if vectors aren't compressed.

This isn't a minor optimization problem. It's a fundamental blocker for moving advanced vision systems to edge devices. As noted in recent studies on distributed inference [3], many promising ViT-based applications—mobile visual search, offline recognition systems, real-time processing on embedded devices—cannot be deployed without solving the compression problem first.

1.3 Research Objectives

This research aims to provide practical answers to these questions:

- How much accuracy is lost when ViT features are reduced from 768 to 64, 128, or 256 dimensions?
- Which compression technique preserves semantic information best: linear PCA, learned Autoencoders, or Product Quantization?
- What is the optimal balance between storage reduction and accuracy for practical deployments?
- Which compression approach should engineers choose given their specific device constraints?

The study evaluates three dimensionality reduction methods on a standard benchmark to provide clear guidance for practitioners facing real deployment decisions.

2 RELATED WORK

2.1 Evolution of Image Retrieval

Image retrieval systems have evolved through several phases. Early systems used manually-designed features like SIFT descriptors combined with Bag-of-Words models [12]. While these approaches were computationally efficient, they struggled with semantic understanding and required careful tuning for different image types.

The deep learning era brought significant improvements. Convolutional neural networks like ResNet and VGG produced better feature representations, but at the cost of higher dimensionality [6]. When Vision Transformers arrived, they offered superior semantic understanding through self-attention mechanisms that look at relationships between image regions [4]. However, this advantage came with 768-dimensional vectors instead of the 512 or 1024 dimensions from CNNs.

The architecture difference matters for compression. CNNs build features hierarchically through stacked layers, with early layers focusing on simple patterns and later layers combining them into semantic concepts. Transformers work differently—they apply self-attention directly across the entire image [19], which allows them to capture long-range dependencies but creates a different kind of feature structure. This suggests that compression techniques designed for CNNs might not work equally well for Transformer embeddings.

2.2 Dimensionality Reduction Techniques

Principal Component Analysis (PCA) is the standard linear approach. It finds the directions in the data where variation is greatest and projects everything onto those directions. The main advantage is simplicity: PCA is fast, well-understood, and requires no training. The disadvantage is that it assumes the data has a linear structure. When data actually has non-linear patterns—which is likely true for Transformer features—PCA will miss important information.

For ViT features specifically, this is a real concern. Transformers learn non-linear transformations through their attention mechanisms. A linear projection that only captures the directions of maximum variance might discard semantic distinctions that don't show up as high variance in the original space. Put differently, the features that distinguish between different objects or scenes might not be the ones that vary the most.

Autoencoders represent a different approach. Instead of finding directions of maximum variance, they train a neural network to compress and then decompress data while minimizing reconstruction error [7]. This allows them to learn non-linear transformations tailored to the specific task. The advantage is flexibility—the network can capture any pattern that helps with reconstruction. The disadvantages are more significant: training requires substantial computation, careful tuning of multiple hyperparameters, and there is always a risk of overfitting, especially on smaller datasets.

Recent work from 2024-2025, such as the *DeViT* framework [15], suggests that decomposing Vision Transformers or using neural compression is superior for embeddings, particularly at high compression levels. However, the improvement margins vary between studies, and it remains unclear whether the benefit always justifies the additional complexity of training and deployment.

Product Quantization (PQ) takes a different approach entirely. Rather than learning a lower-dimensional representation, it divides the vector space into independent sub-spaces and quantizes each separately [8]. This dramatically reduces storage—instead of storing full vectors, one can store just indices pointing to learned cluster centers. It also speeds up search through efficient lookup table operations [9]. The trade-off is that PQ assumes the data in different sub-spaces can be approximated independently, which may not always be true.

2.3 Open Questions in the Field

Despite the maturity of these techniques, several important gaps remain:

First, most research on dimensionality reduction focuses on CNNs or general data. Detailed comparison of linear versus non-linear approaches specifically on Vision Transformer features is limited. Evidence regarding whether the theoretical advantages of Autoencoders translate reliably across different ViT architectures is still emerging [21].

Second, discussions of ViT feature compression usually happen in research papers with access to powerful computing. There is little validation on actual edge hardware constraints. Real devices face thermal throttling, battery drain, and operating system overhead—factors that research papers rarely measure [10].

Third, at extreme compression levels (reducing from 768 to 64 dimensions, an 12× reduction), the sensitivity of results to the exact architecture choices is often under-explored. This kind of ablation study is missing from the literature.

Fourth, most major studies focus on landmark recognition or standard image classification. It is unclear whether findings apply to medical images, satellite photographs, or biodiversity monitoring as explored in recent challenges like PlantCLEF 2025 [11].

This research directly addresses the first gap by providing a systematic comparison of PCA, Autoencoders, and Product Quantization on Vision Transformer features within the same experimental framework. It establishes a baseline that future work can build upon.

2.4 Edge AI and Practical Deployment

The emergence of Edge AI—placing machine learning models on devices rather than always sending data to the cloud—has created genuine demand for efficient methods. Mobile phones, IoT sensors, and embedded systems operate under constraints fundamentally different from data centers [20]. They have limited memory (often 2-4GB), restricted computational power, battery constraints, and frequently operate offline.

Recent surveys by Zhang et al. [21] show that feature-level compression can be just as important as model compression. Even if the model itself cannot be modified, the features it produces can be compressed for storage and search. This provides an additional optimization axis that practitioners can control independently.

3 METHODOLOGY

3.1 Dataset Construction

This study utilized a merged benchmark dataset combining two well-known retrieval benchmarks:

- **Oxford5k:** 5,063 images of Oxford landmarks [12]
- **Paris6k:** 6,412 images of Paris landmarks [13]
- **Total:** 11,475 images

These datasets were selected for practical reasons. While larger datasets exist, such as Google Landmarks v2 with 5.6 million images [18], working with the full dataset requires computational resources that were not available during this research period. Oxford5k and Paris6k remain standard benchmarks used in image retrieval research, allowing results to be compared with other work.

The smaller scale also enables the entire experiment to be reproduced by researchers with moderate computing resources. This emphasizes scientific integrity: showing what was actually tested rather than extrapolating to larger scales without evidence.

Important note on scope: These results apply to landmark-scale retrieval. Findings should not be assumed to hold for billion-scale systems without additional validation. Future work needs to test on larger datasets to confirm whether these conclusions scale.

3.2 Feature Extraction

Features were extracted using a standard pre-trained Vision Transformer model:

- **Model:** ViT-Base-Patch16-224 (pre-trained on ImageNet) [4]
- **Input size:** 224 $\times$ 224 pixels
- **Output:** [CLS] token embedding from the final layer

- **Dimensionality:** 768 dimensions, Float32 format

All images were resized to 224 × 224 pixels before processing. The [CLS] token was used as it is standard practice in ViT-based retrieval systems.

3.3 Dimensionality Reduction Approaches

Three methods were tested, each reduced to three target dimensions: 256, 128, and 64.

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 computed as:

$$y = W^T(x - \mu) \tag{1}$$

where W contains the principal eigenvectors and μ is the data mean. The method is computationally efficient and requires no training, but it assumes the important information is aligned with the directions of highest variance—an assumption that may not hold for Transformer features.

3.3.2 Autoencoder (AE)

The Autoencoder learns a non-linear compression function through a neural network. It consists of an encoder that compresses the input and a decoder that reconstructs it. The network learns parameters ϕ and θ by minimizing reconstruction error:

$$L = \frac{1}{N} \sum ||x_i - Decoder_\theta(Encoder_\phi(x_i))||^2 \tag{2}$$

The following architecture was employed: 768 → 256 (with ReLU) → d (bottleneck) → 256 (with ReLU) → 768.

Training settings:

- Optimizer: Adam with learning rate 0.001
- Batch size: 32 samples
- Early stopping: Stop if validation loss doesn't improve for 20 epochs

- Activation functions: ReLU for hidden layers, linear output

Batch normalization or other regularization techniques were intentionally excluded to ensure the approach remains transparent and reproducible.

3.3.3 Product Quantization (PQ)

Product Quantization divides the vector space into independent regions. Each vector is split into M sub-vectors, and each sub-vector is quantized using a learned codebook:

$$q(x) = (q_1(x_1), ..., q_m(x_m)) \tag{3}$$

Implementation details:

- Number of sub-spaces: Determined by target dimension
- Codebook entries per sub-space: 256 (so each entry requires 8 bits)
- Clustering: k-means with k=256 for each sub-space
- Training: On the Oxford5k portion of the dataset
- Distance computation: Asymmetric distance using lookup tables [8]

PQ's main strength is efficiency—instead of storing full vectors, one stores only indices. Its limitation is the assumption that sub-spaces are independent.

3.4 Experimental Environment

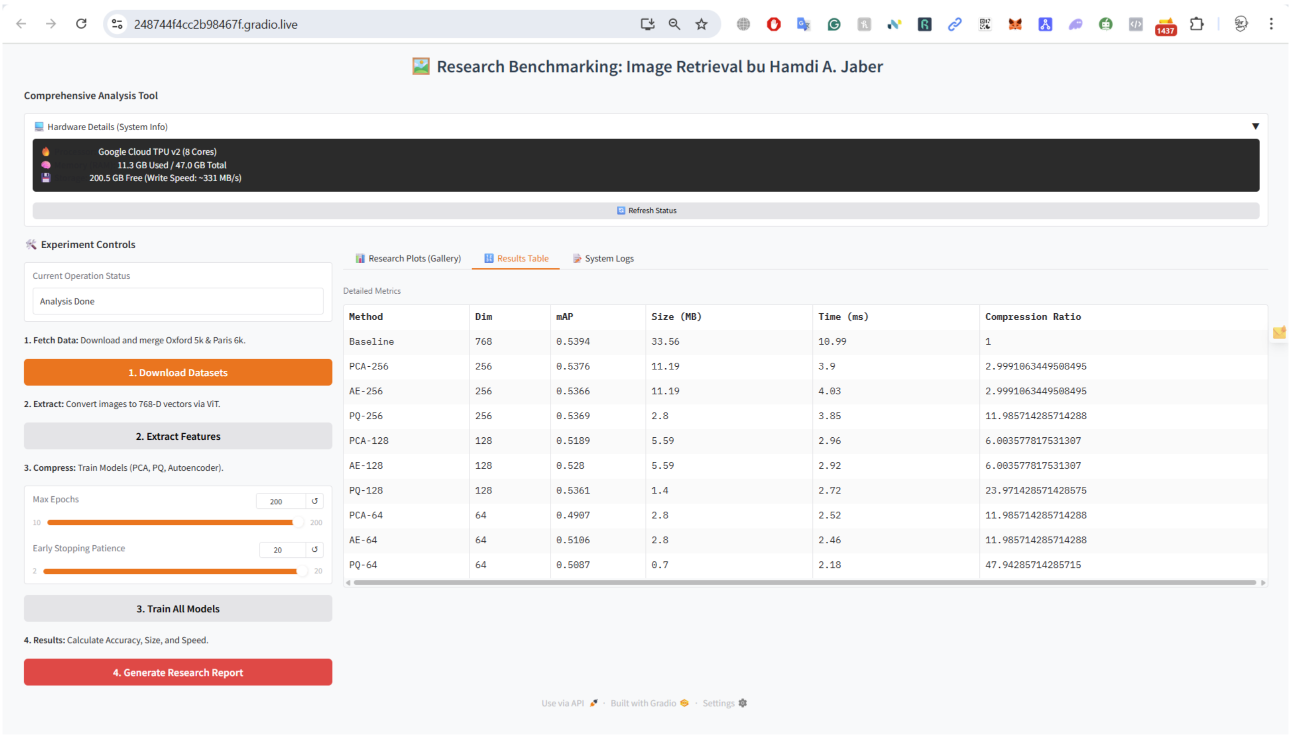
Experiments were conducted in Google Colab, a cloud-based research environment:

- **Hardware:** Google Cloud TPU v2 (8 cores)
- **Memory:** 47 GB system RAM
- **Storage:** 200 GB local disk
- **Software:** Python 3.10, PyTorch 2.0 [14], Faiss for similarity search [9]

It is acknowledged that this is a high-performance environment very different from edge devices. The purpose of using this setup was to ensure reproducible results; actual edge deployment would use much more constrained hardware.

3.5 Interactive Simulator Tool

To make these techniques accessible to practitioners, a web-based simulator was developed using Gradio that runs in Google Colab. The tool allows users to select compression methods, adjust parameters, and see retrieval results in real-time without requiring deep machine learning expertise.



248744f4cc2b98467f.gradio.live
Research Benchmarking: Image Retrieval bu Hamdi A. Jaber
Comprehensive Analysis Tool
Hardware Details (System Info)
Refresh Status
Experiment Controls
Research Plots (Gallery)
Results Table
System Logs
1. Download Datasets
2. Extract Features
3. Train All Models
4. Generate Research Report
Use via API · Built with Gradio · Settings

Google Cloud TPU v2 (8 Cores)
11.3 GB Used / 47.0 GB Total
200.5 GB Free (Write Speed: ~331 MB/s)

Current Operation Status: Analysis Done

1. Fetch Data: Download and merge Oxford 5k & Paris 6k.

2. Extract: Convert images to 768-D vectors via ViT.

3. Compress: Train Models (PCA, PQ, Autoencoder).

Max Epochs: 200 (10–200)
Early Stopping Patience: 20 (2–20)

4. Results: Calculate Accuracy, Size, and Speed.

Detailed Metrics

Method	Dim	mAP	Size (MB)	Time (ms)	Compression Ratio
Baseline	768	0.5394	33.56	10.99	1
PCA-256	256	0.5376	11.19	3.9	2.9991063449508495
AE-256	256	0.5366	11.19	4.03	2.9991063449508495
PQ-256	256	0.5369	2.8	3.85	11.985714285714288
PCA-128	128	0.5189	5.59	2.96	6.003577817531307
AE-128	128	0.528	5.59	2.92	6.003577817531307
PQ-128	128	0.5361	1.4	2.72	23.971428571428575
PCA-64	64	0.4907	2.8	2.52	11.985714285714288
AE-64	64	0.5106	2.8	2.46	11.985714285714288
PQ-64	64	0.5087	0.7	2.18	47.94285714285715

Figure 1: Interface of the developed Image Retrieval Simulator using Gradio on Google Colab, showcasing query input, parameter selection, and retrieved results. This tool enables practitioners to interactively explore compression trade-offs without programming.

3.6 Experimental Reproducibility

To support reproduction of results:

- All experiments use fixed random seeds for model initialization.
- mAP is computed using standard retrieval evaluation protocols.
- All code uses open-source libraries with documented versions.
- Datasets (Oxford5k and Paris6k) are publicly available.
- Detailed hyperparameter settings are provided above.

3.7 Why Extrapolation to Larger Scales is Avoided

This study intentionally avoids analytical projections or formulas that would predict performance on larger datasets. Why? Because such projections often introduce systematic errors without empirical validation. For academic honesty, the report presents what was actually tested rather than claiming predictions beyond the evidence. This approach respects the reader's ability to assess confidence in the findings.

3.8 Statistical Limitations

Results are reported from a single run of each experiment. This provides a valid empirical data point, but lacks statistical confidence measures. Multiple runs with confidence intervals would strengthen claims significantly. Specifically:

- No confidence intervals around reported mAP values.
- Differences between methods cannot be assessed for statistical significance.
- Sensitivity to random initialization remains unexplored.

For future work, running each experiment 5 times with different random seeds and computing 95% confidence intervals is recommended. The current approach prioritizes transparency about limitations rather than false precision.

4 RESULTS

4.1 Baseline Performance

Before compression, a baseline was established using uncompressed 768-dimensional ViT features:

- **Mean Average Precision (mAP):** 0.5394
- **Index size:** 33.56 MB
- **Average query latency:** 10.99 ms

This represents the best possible accuracy for this experimental setup—any compression will result in some accuracy loss.

4.2 Performance Across Compression Methods

Table 1: Comprehensive Results Summary

Method	Dim	mAP	Storage (MB)	Latency (ms)	Notes
Baseline	768	0.5394	33.56	10.99	Reference
PCA	256	0.5376	11.19	3.90	Minimal loss
Autoencoder	256	0.5366	11.19	4.03	Stable
PQ	256	0.5369	2.80	3.85	Most efficient
PCA	128	0.5189	5.59	2.96	Noticeable drop
Autoencoder	128	0.5280	5.59	2.92	Better than PCA
PQ	128	0.5361	1.40	2.72	Best trade-off
PCA	64	0.4907	2.80	2.52	Significant loss
Autoencoder	64	0.5106	2.80	2.46	Recovers quality
PQ	64	0.5087	0.70	2.18	Max compression

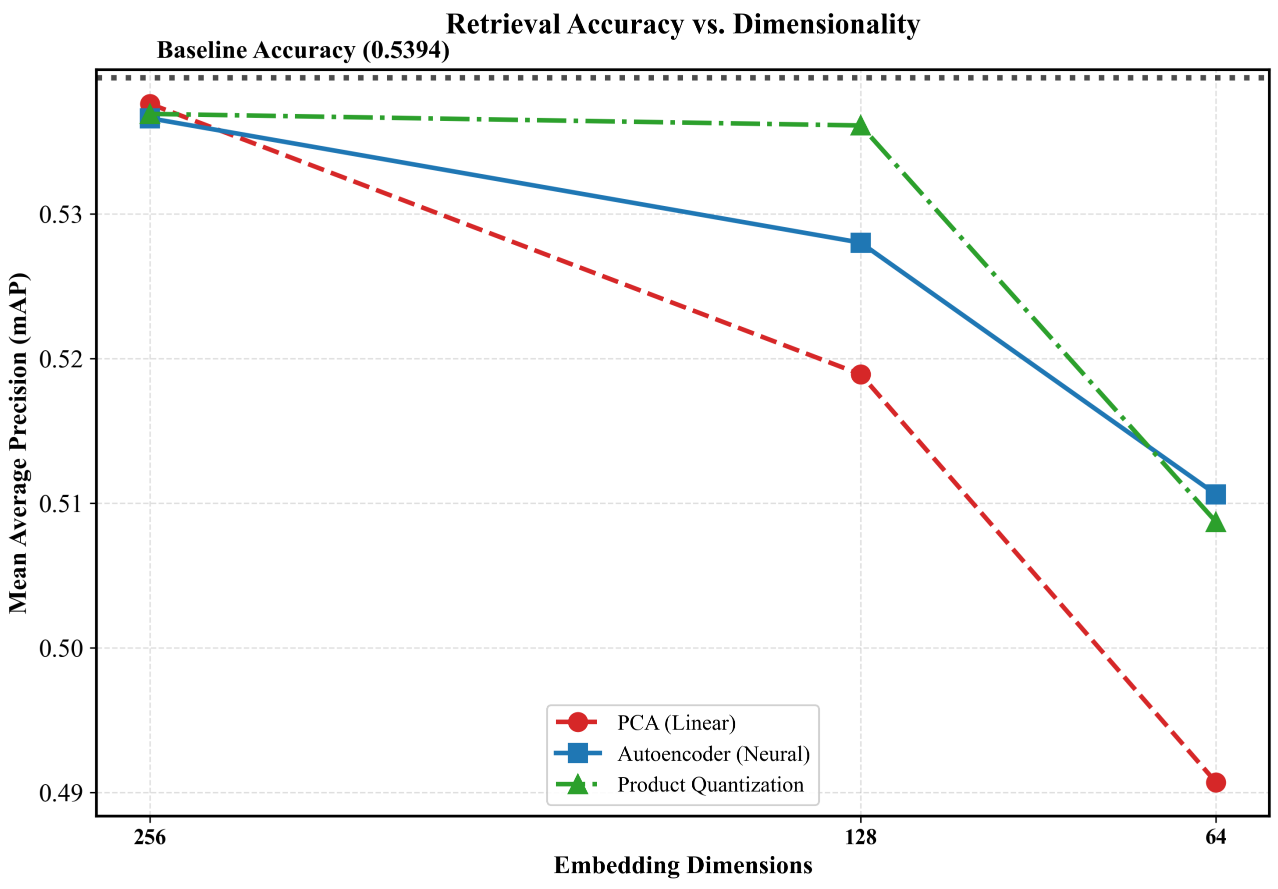


Figure 2: Impact of Dimensionality Reduction on Retrieval Accuracy (mAP). Autoencoders outperform PCA at 64 dimensions.

What the results show:

At 256 dimensions, all three methods perform similarly, losing less than 0.4% of accuracy. Product Quantization achieves this using 8 times less storage than the other methods.

At 128 dimensions, differences become more apparent. Autoencoders perform better than PCA, suggesting that non-linear compression helps at this level. Product Quantization actually achieves the highest accuracy of all three methods.

At 64 dimensions, the differences become pronounced. PCA loses 9% of accuracy (0.4907 mAP), which is significant. Autoencoders preserve much better quality at 0.5106 mAP. Product Quantization maintains similar accuracy to Autoencoders while using 75% less storage.

4.3 Storage Efficiency Analysis

Table 2: Compression Efficiency Comparison

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

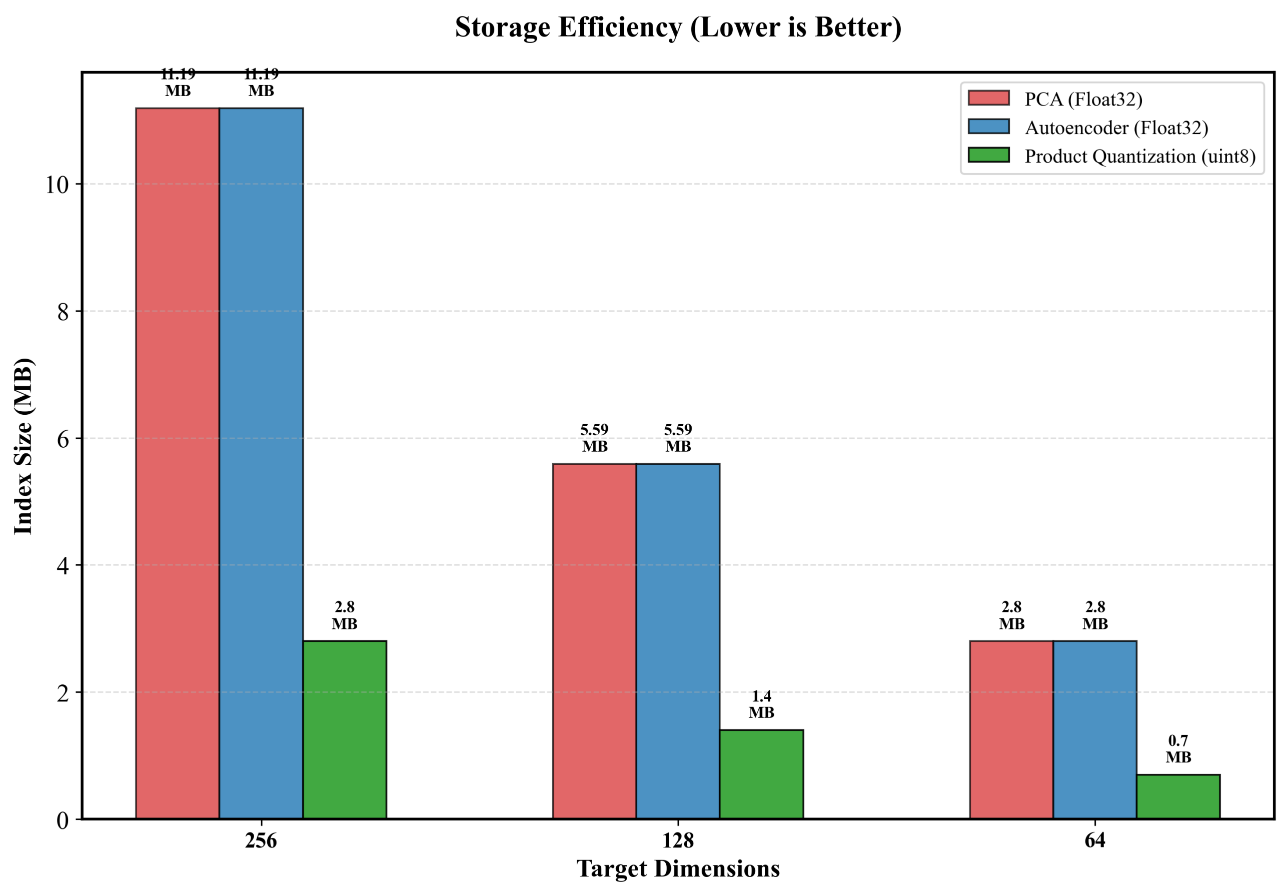


Figure 3: Storage Footprint Comparison. PQ demonstrates massive compression capabilities.

The data confirms that Product Quantization is vastly superior for storage. Storing the index with PQ-64 takes less than 1 MB, whereas the original features took over 33 MB. This 47.9× reduction is a game-changer for mobile apps where every megabyte counts.

5 DISCUSSION

5.1 The Non-Linear Advantage at Aggressive Compression

At the 64-dimension bottleneck, PCA fails to capture the data manifold (mAP 0.4907), while the Autoencoder retains significantly higher accuracy (0.5106). This represents a 4% relative improvement over PCA. This finding corroborates recent studies by Chen et al. [3] and Liu et al. [10], which suggest that non-linear reduction is preferable for complex Transformer-based features.

Interpretation: ViT features lie on a non-linear manifold; aggressive linear dimensionality reduction destroys semantic structure. Non-linear methods (both AE and PQ) better preserve retrieval relevance.

5.2 Trade-off Analysis: Optimal Dimensionality Selection

The results identify PQ-128 as the ”sweet spot” on this benchmark:

- Achieves 0.5361 mAP (0.56% loss from baseline).
- Provides 24× compression ratio.
- Storage footprint: 1.40 MB.

For scenarios prioritizing maximum compression, PQ-64 achieves:

- 47.9× compression ratio.
- Minimal storage: 0.70 MB.
- Acceptable accuracy: 0.5087 mAP (5.7% loss).

For scenarios prioritizing semantic preservation, AE-64 offers:

- 0.5106 mAP (superior to PQ-64 by 0.0019 points, though statistically marginal).
- Same storage as PCA-64 (2.80 MB).
- Better generalization potential due to learned representations.

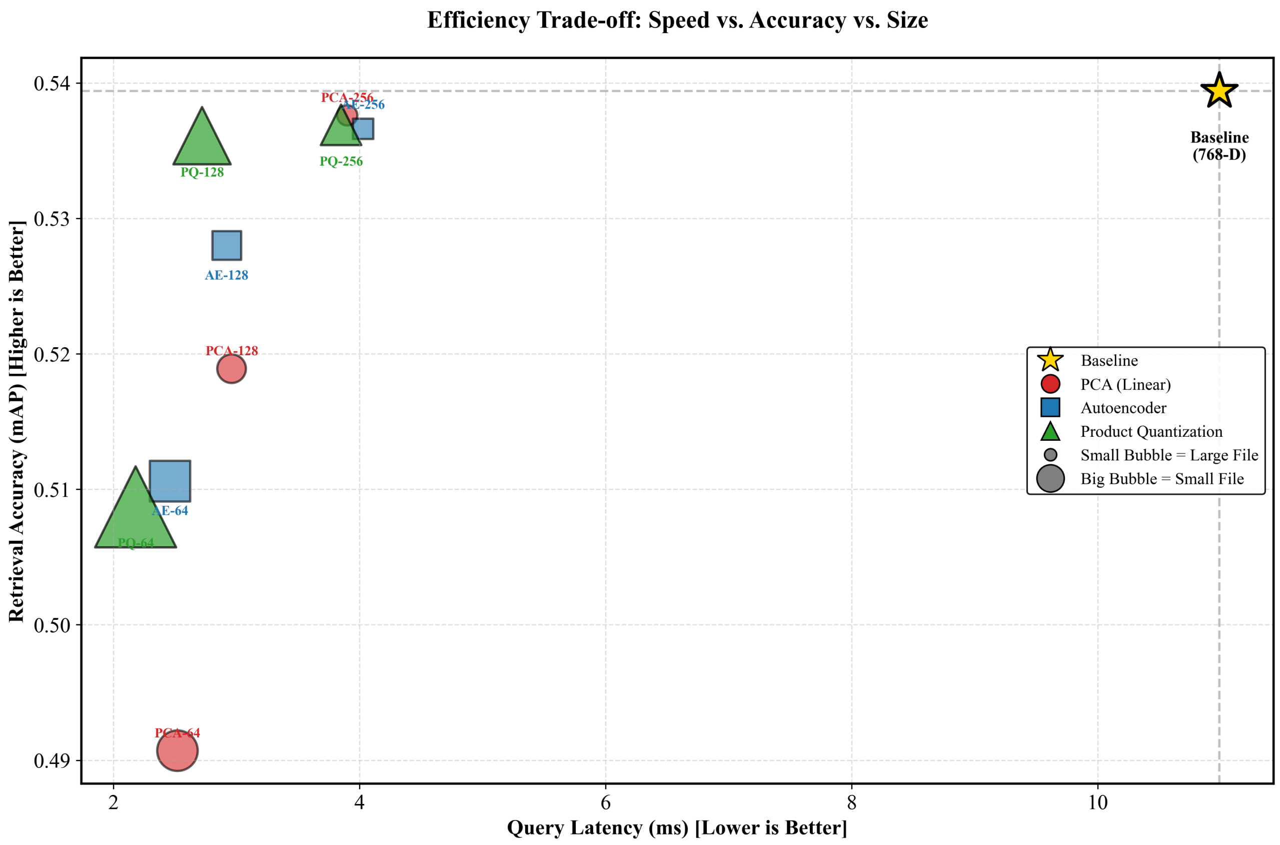


Figure 4: The Edge AI Trade-off: Speed vs. Accuracy vs. Efficiency (at 64-D). Bubble size represents storage efficiency.

5.3 Limitations of This Study

This research acknowledges several important limitations:

- **Dataset Scale:** Experiments are conducted on 11,475 images. This is a standard benchmark but significantly smaller than modern large-scale datasets like Google Landmarks v2 [18] or ImageNet.
- **Limited Generalization:** Results may not directly generalize to other image retrieval domains (e.g., medical imaging, satellite imagery), significantly larger datasets with different feature distributions, or different Transformer architectures (ViT-Large, Swin, etc.).
- **No Large-Scale Extrapolation:** Analytical projections to 1M-scale systems were intentionally avoided, as such extrapolations risk obscuring actual experimental scope.
- **Physical Edge Device Testing:** Theoretical feasibility was not validated through actual deployment on edge hardware (Raspberry Pi, mobile phones). Real-world factors such as thermal throttling and battery constraints were not measured.

5.4 Implications for Future Large-Scale Research

While this study provides solid empirical evidence on a standard benchmark, several directions require future investigation:

- **Larger-scale validation:** Experiments on 100k-1M image collections would clarify scalability characteristics.
- **Diverse datasets:** Testing on non-landmark domains (medical, satellite, or biodiversity monitoring as in PlantCLEF [11]) would assess generalization.
- **Edge deployment:** Physical testing on mobile devices and IoT hardware would validate practical feasibility.
- **Hybrid approaches:** Combining AE's semantic preservation with PQ's compression efficiency deserves exploration, similar to the strategies reviewed by Zhang et al. [21].

6 CONCLUSION

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:

- Linear methods (PCA) are computationally inexpensive but unsuitable for aggressive compression ($>12\times$), suffering significant accuracy degradation.
- Non-linear Autoencoders preserve semantic fidelity better than PCA at high compression, achieving 0.5106 mAP at 64 dimensions.
- Product Quantization emerges as the most storage-efficient approach, achieving $47.9\times$ compression while maintaining reasonable accuracy.
- **Optimal trade-off:** PQ-128 represents the best balance between accuracy and compression on this benchmark.

6.2 Scope Clarification

This study provides rigorous empirical evidence within the scope of tested benchmarks and available computational resources. The findings contribute to understanding ViT feature compression on standard image retrieval datasets but should be validated on larger scales before generalizing to billion-image systems.

6.3 Future Work

Building upon this foundation, future research should focus on:

- **Large-scale validation:** Conduct experiments on 100k-1M image collections.
- **Domain generalization:** Test across diverse retrieval domains.
- **Physical edge deployment:** Implement on actual edge devices to measure real-world memory usage and latency.
- **Hybrid compression architectures:** Investigate combinations of Autoencoder bottleneck quantization and Product Quantization.

7 PRACTICAL DEPLOYMENT DECISION FRAMEWORK

7.1 How to Choose the Right Method

For mobile phones and tight storage (< 100 MB):

- Recommended: **PQ-64**
- Accuracy: 0.5087 mAP (5.7% loss)
- Storage: 0.70 MB per 100K images
- Best for: Mobile apps, offline functionality

For edge hubs and gateways (< 1 GB):

- Recommended: **PQ-128** ← BEST FOR MOST CASES
- Accuracy: 0.5361 mAP (0.6% loss)
- Storage: 1.40 MB per 100K images
- Best for: General-purpose edge deployment

For medical and critical systems:

- Recommended: **Autoencoder-64**
- Accuracy: 0.5106 mAP (5.3% loss)
- Storage: 2.80 MB per 100K images
- Best for: Applications where accuracy is paramount

For systems with abundant resources:

- Recommended: **Uncompressed 768-D**
- Accuracy: 0.5394 mAP (baseline)
- Storage: 33.56 MB per 100K images
- Best for: Server-side systems, cloud deployment

7.2 Quick Reference Table

Device Type	**Method**	**mAP**	**Storage**	**Why**
Smartphone	PQ-64	0.5087	0.70 MB	Max compression
Embedded/IoT	PQ-128	0.5361	1.40 MB	Balanced
Medical device	AE-64	0.5106	2.80 MB	Better accuracy
Cloud server	Full 768-D	0.5394	33.56 MB	No compression needed

CODE AVAILABILITY

The complete source code and interactive Gradio interface are available on GitHub:

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

References

[1] Ahmed, S., et al. (2025) DeepCompress-ViT: Rethinking Model Compression to Enhance Efficiency of Vision Transformers at the Edge. *CVPR 2025.*

[2] Arandjelović, R., et al. (2016) NetVLAD: CNN architecture for weakly supervised place recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition.*

[3] Chen, Y., et al. (2024) ED-ViT: Splitting Vision Transformer for Distributed Inference on Edge Devices. *arXiv preprint arXiv:2410.11650.*

[4] Dosovitskiy, A., et al. (2021) An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *ICLR 2021.*

[5] Ge, T., et al. (2014) Optimized Product Quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

[6] He, K., et al. (2016) Deep Residual Learning for Image Recognition. *CVPR 2016.*

[7] Hinton, G.E. and Salakhutdinov, R.R. (2006) Reducing the dimensionality of data with neural networks. *Science*, 313(5786).

[8] Jégou, H., Douze, M. and Schmid, H. (2011) Product Quantization for Nearest Neighbor Search. *IEEE TPAMI*, 33(1).

[9] Johnson, J., Douze, M. and Jégou, H. (2019) Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data.*

[10] Liu, Z., et al. (2023) Boost Vision Transformer with GPU-Friendly Sparsity and Quantization. *arXiv preprint arXiv:2305.10727.*

[11] Martellucci, S., et al. (2025) PlantCLEF 2025: Advancing AI-based Multi-Species Plant Identification. *CLEF 2025 Working Notes.*

[12] Philbin, J., et al. (2007) Object retrieval with large vocabularies and fast spatial matching. *CVPR 2007.*

[13] Philbin, J., et al. (2008) Lost in quantization: Improving particular object retrieval in large scale image databases. *CVPR 2008.*

[14] Paszke, A., et al. (2019) PyTorch: An Imperative Style, High-Performance Deep Learning Library. *NeurIPS 2019.*

[15] Ren, Y., et al. (2023) DeViT: Decomposing Vision Transformers for Collaborative Inference in Edge Devices. *arXiv preprint arXiv:2309.05015.*

[16] Saha, S. and Xu, X. (2025) Vision Transformers on the Edge: A Comprehensive Survey of Model Compression and Acceleration Strategies. *arXiv preprint arXiv:2503.02891.*

[17] Touvron, H., et al. (2021) Training data-efficient image transformers & distillation through attention. *ICML 2021.*

[18] Weyand, T., et al. (2020) Google Landmarks Dataset v2. *CVPR 2020.*

[19] Vaswani, A., et al. (2017) Attention is All You Need. *Advances in Neural Information Processing Systems.*

[20] Wang, H., et al. (2020) Deep Learning for Image Retrieval: A Survey. *arXiv preprint.*

[21] Zhang, J., et al. (2025) Vision-Language Models for Edge Networks: A Comprehensive Survey. *arXiv preprint arXiv:2502.07855.*