Fast Image Vector Quantization Using Sparse Oblique Regression Trees

Rasul Kairgeldin Miguel Á. Carreira-Perpiñán

Dept. Computer Science & Engineering
University of California, Merced

September 5, 2025

Vector Quantization (VQ) and Image Coding

- VQ is widely used for image coding, representing image patches as vectors in Euclidean space.
- Each image patch is approximated using a finite set of vectors (the codebook).
- Can be used to compress images by replacing high-dimensional patches with compact codebook indices.
- Codebooks are typically learned by minimizing squared Euclidean distortion.
- The k-means algorithm is the standard method to optimize the codebook:
  - Minimizes squared Euclidean distance (distortion) between patches and assigned codebook vectors
  - Alternates between assigning patches to codebook vectors and updating the codebook vectors.
  - Limitation: encoding cost is linear in K (number of codebook vectors), as each test patch must be compared to all K vectors (slow for large K).
  - Used as a baseline in comparisons with advanced VQ methods.

Tree-Structured VQ (TSVQ) for Efficient Encoding

- Encoding can be sped up using a tree-structured codebook:
  - Only one root-to-leaf path is traversed (logarithmic time in K)
  - Enables use of large codebooks without incurring prohibitive encoding cost
- Uses a binary tree (oblique or univariate) of depth Δ. In previous work, this was learned by greedy recursive partitioning:
  - Each decision node applies a hyperplane split
  - Each leaf stores a constant codebook vector (or assignment)
- The tree partitions space into convex polytopes, unlike standard Voronoi partitions
- Two major challenges:
  - Designing decision nodes that balance flexibility and computational cost
  - Learning such trees is nonconvex and nondifferentiable

Related Work

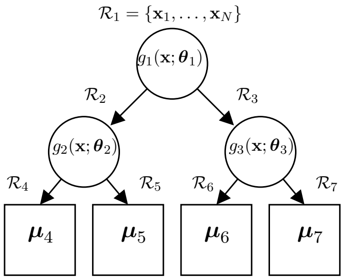
- Traditional regression trees (CART [1] or C5.0 [10]):
  - Use greedy recursive partitioning (do not optimize a global loss function)
  - Typically axis-aligned
  - Result in large, often inaccurate trees—poor in high-dimensional settings
- Tree-Structured Vector Quantization (TSVQ):
  - Builds codebooks hierarchically for fast encoding (logarithmic in codebook size)
  - Commonly built with greedy [7, 5, 8] or random recursive partitioning [4]
  - Often pruned post-training to improve rate-distortion performance [3, 9]
- Segmentation-based coding in image compression [11, 9, 6, 8]:
  - Applies quantization to image segments for adaptive encoding
  - Useful in medical image compression and other visual domains
- Limitations of previous tree-based VQ:
  - Prior methods often lack global optimization
  - May require deeper trees to achieve acceptable distortion levels

Summary of Our Contribution

- Propose a tree-structured VQ model using sparse oblique regression trees for the first time so that they optimize the squared distortion properly
- Propose a hierarchical formulation of VQ:
  - Replaces flat codebook assignments with a structure implicitly defined by a decision tree
- Apply Tree Alternating Optimization (TAO) [2] to train VQ models for the first time:
  - Jointly learns both the sparse oblique hyperplane splits at internal tree nodes and the codebook vectors in the leaves
- Demonstrate that the method achieves:
  - Lower distortion compared to other TSVQ methods (close to k-means)
  - Faster encoding time, especially for large codebooks

VQ with Sparse Oblique Decision Trees

- Binary tree of depth Δ
- Decision nodes $\mathcal{D}$:
  - Each contains a routing function $g_i(\mathbf{x};\boldsymbol{\theta}_i)\colon \mathbb{R}^D \to \{\texttt{left}_i, \texttt{right}_i\} \subset \{\mathcal{D} \cup \mathcal{L}\}$
  - $g_i(\mathbf{x};\boldsymbol{\theta}_i) = \texttt{left}_i$ if $\mathbf{w}_i^T\mathbf{x} + w_{0i} < 0$, otherwise $\texttt{right}_i$
- Leaf nodes $\mathcal{L}$:
  - Each contains codebook vector $\boldsymbol{\mu}_j \in \mathbb{R}^D$
- Tree's learnable parameters: $\boldsymbol{\Theta} = \{(\mathbf{w}_i, w_{0i})\}_{i\in\mathcal{D}} \cup \{\boldsymbol{\mu}_j\}_{j\in\mathcal{L}}$
- Tree routing function $\mathbf{T}(\mathbf{x}_n;\boldsymbol{\Theta})$ directs a patch $\mathbf{x}_n$ from a root to a single leaf and predicts a corresponding codeword $\boldsymbol{\mu}_j$



TSVQ Problem Formulation

- We formulate the tree-structured VQ problem as follows:

$$\min_{\boldsymbol{\Theta}} \sum_{n=1}^{N} \left\| \mathbf{x}_n - \mathbf{T}(\mathbf{x}_n; \boldsymbol{\Theta}) \right\|^2 + \lambda \sum_{i \in \mathcal{D}} \|\mathbf{w}_i\|_1 \tag{1}$$

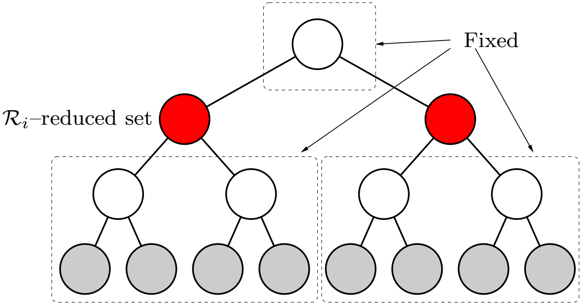
- Problem (1) can be seen as generalizing the regular squared distortion over a dataset of patches $\{\mathbf{x}_n\}_{n=1}^{N} \subset \mathbb{R}^D$ from a flat codebook to a hierarchical one
- Assignments are not free variables, but implicitly determined by the tree structure (not a Voronoi cell, but a trainable polytope)
- The codebook consists of the leaf node vectors
- Hyperparameters:
  - Tree depth Δ controls primary model capacity and codebook size
  - Sparsity parameter λ can prune the tree by zeroing out weights, reducing complexity (secondary control of codebook size)

Tree Alternating Optimization (TAO): Separability Condition

- TAO is used to optimize objective function (eq (1))
- TAO iteratively updates each node's parameters so as to decrease the objective function
- TAO algorithm is based on 2 theorem: separability condition and reduced problem over nodes.

TAO: Separability Condition

Assume the parameters are not shared across nodes ($i \neq j \Rightarrow \boldsymbol{\theta}_i \cap \boldsymbol{\theta}_j = \emptyset$). Assume nodes i and j are non-descendant of each other, and all other parameters ($\boldsymbol{\Theta}_{rest} = \boldsymbol{\Theta} \backslash \{\boldsymbol{\theta}_i, \boldsymbol{\theta}_j\}$) are fixed.



TAO: Separability Condition

In general, if $\mathcal{S} \subset \mathcal{N}$ is a nonempty set of non-descendant nodes in the tree and $\{\boldsymbol{\theta}_i : i \in \mathcal{S}\}$ is the set of their parameters, then $E(\boldsymbol{\Theta})$ can be rewritten as:

$$E(\boldsymbol{\Theta}) = \sum_{i \in \mathcal{S}} E_i(\boldsymbol{\theta}_i) + E_{\text{rest}}(\boldsymbol{\Theta}_{\text{rest}})$$

Optimization of $i \in \mathcal{S}$ can be done in parallel, which drastically facilitates the process.

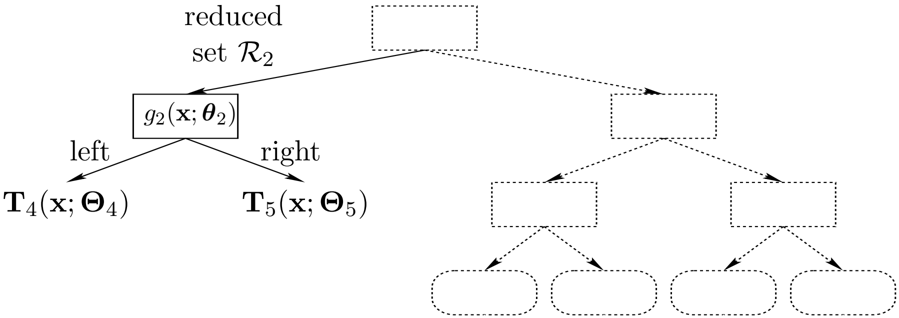
TAO: Reduced Problem Over Leaves

Reduced problem over leaves states that for a leaf $j \in \mathcal{L}$ the problem (1) is reduced to an original loss (squared distortion) between the codeword of a leaf $\boldsymbol{\mu}_j$ and the leaf's reduced set $\mathcal{R}_j$. It can be solved by finding a mean (similar to k-means):

$$\min_{\boldsymbol{\mu}_j} \sum_{n \in \mathcal{R}_j} \left\| \mathbf{x}_n - \boldsymbol{\mu}_j \right\|^2 \tag{2}$$

The solution is $\boldsymbol{\mu}_j = \frac{1}{|\mathcal{R}_j|} \sum_{n \in \mathcal{R}_j} \mathbf{x}_n$

TAO: Reduced Problem Over Decision Nodes



- $\mathcal{R}_i$ is the reduced set of decision node i.
- Function $l_{in} : \mathcal{C}_i \rightarrow \mathbb{R}$ as $l_{in} = L(\mathbf{y}_n, \mathbf{T}_z(\mathbf{x}_n;\boldsymbol{\Theta}_z))$, for any $z \in \mathcal{C}_i$ (child of i).

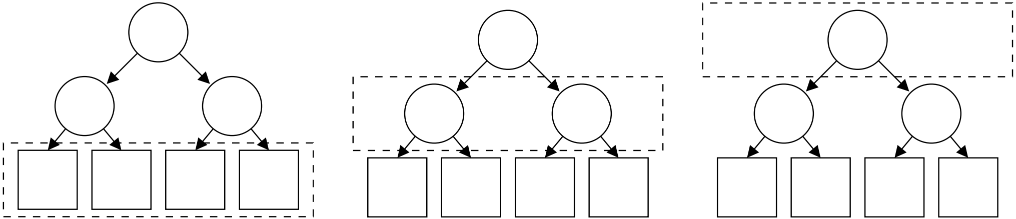
TAO: Reduced Problem Over Decision Nodes

This is NP-hard to optimize, but it can be approximated by a convex surrogate (we use a weighted ℓ_1-regularized logistic regression classifier):

$$\min_{\boldsymbol{\theta}_i} \sum_{n \in \mathcal{R}_i} \overline{L}(\overline{y}_n, g_i(\mathbf{x}_n; \boldsymbol{\theta}_i)) + \lambda \left\| \mathbf{w}_i \right\|_1 \tag{3}$$

where a weighted 0-1 loss $\overline{L}_{in}(\overline{y}_{in}, \cdot)$ of each sample $n \in \mathcal{R}_i$ is defined as $\overline{L}_{in}(\overline{y}_{in}, y) = l_{in}(y) - l_{in}(\overline{y}_{in})\ \forall y \in \{\texttt{left}, \texttt{right}\}$, where $\overline{y}_{in} = \arg\min_y l_{in}(y)$ is a "pseudolabel" indicating the optimal child to route to.

TAO: Optimization



At each TAO teration, nodes of the same depth are trained in parallel and the algorithm proceeds in the reverse BFS order.

Computational Complexity: Training and Encoding

Training complexity for a complete tree of depth Δ and K leaves on a dataset $\{\mathbf{x}_n\}_{n=1}^{N} \subset \mathbb{R}^D$:

- Root node: $\mathcal{O}(\Delta DN) + \mathcal{O}(cDN)$(pseudolabel assignment and binary classifier fitting with c iterations)
- Decision nodes at level Δ_i can be optimized in parallel, the complexity is: $\mathcal{O}(\Delta_i DN) + \mathcal{O}(cDN)$
- Total complexity at most $\mathcal{O}(\Delta^2 DN) + \mathcal{O}(c\Delta DN)$

Training complexity comparison to k-means for large K:

- Decision node training is $\approx \mathcal{O}(DN \log^2 K)$, much faster than k-means' $\mathcal{O}(DNK)$
- Leaf optimization matches k-means' centroid step at $\mathcal{O}(ND)$

Test patch encoding is $\mathcal{O}(D \log K)$ for a TSVQ (root-to-leaf path) versus $\mathcal{O}(DK)$ for k-means

Experiments

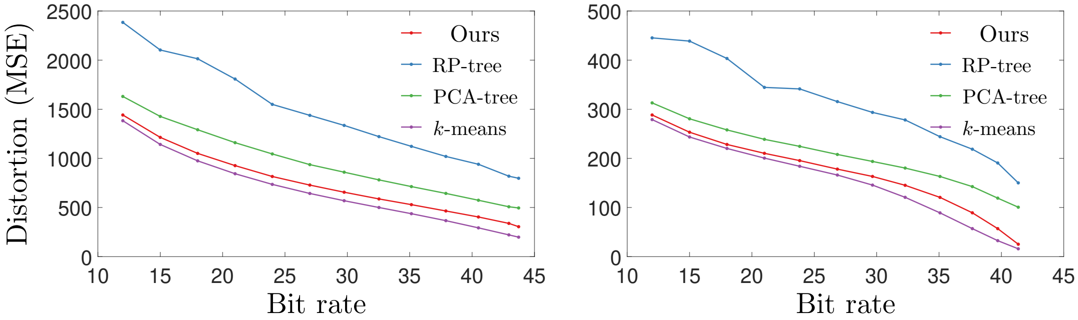


Figure: Distortion (MSE) vs bit rate (Rate-Distortion curve) for different patch size (from left to right 5×5 and 15×15) of each method (TAO-tree, k-means, PCA-tree and RP-tree) on Kodak dataset.

Experiments: Encoding Time and FLOPs vs Distortion

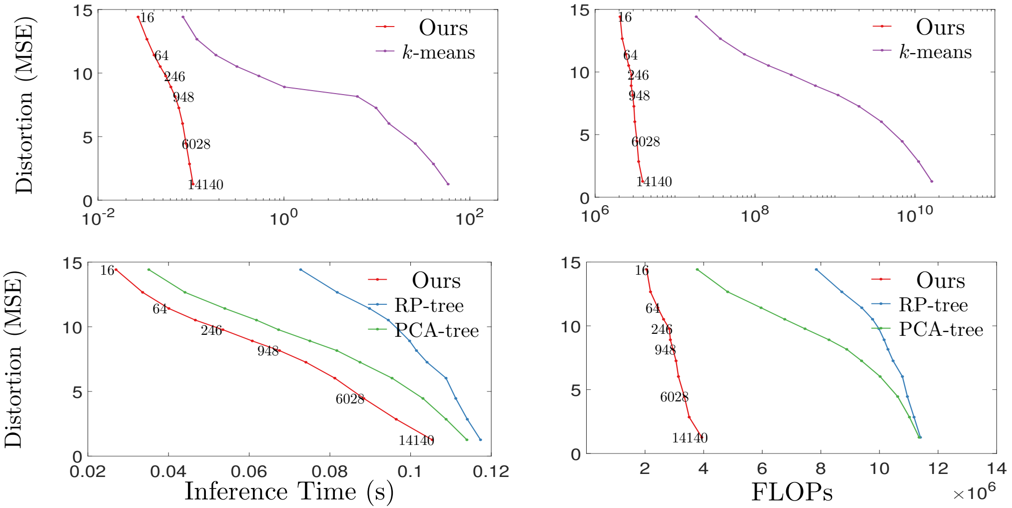


Figure: For given MSE proposed approach achieves much faster encoding time

Experiments: Quantization Quality with 10×10 Patch

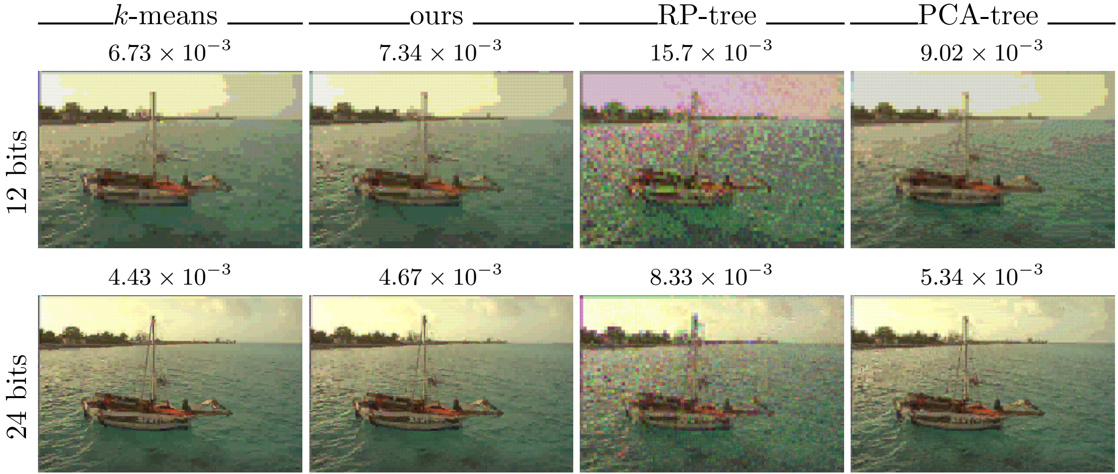


Figure: Distortion (MSE) and bit rate is on top of each image. The proposed method, on par with k-means, displays the best image quality

Experiments: Comparison to k-means for Different Patch Size

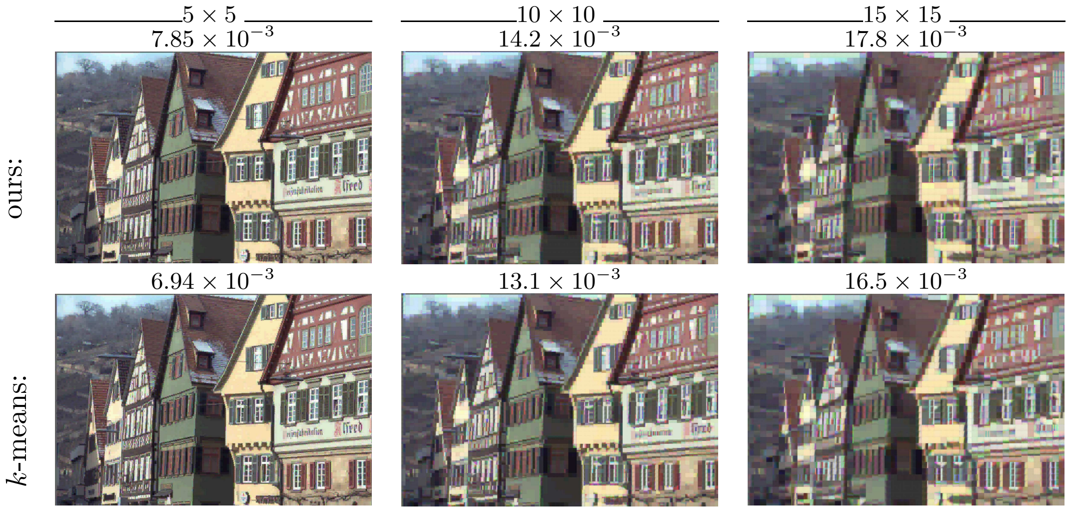


Figure: Quantization quality produced by k-means and our method for 21 bit encoding

Conclusion

A good image patch encoder should:

- Flexibly partition input space for low distortion
- Enable fast decoding
- Be learnable from data

The proposed method:

- A new vector quantizer based on sparse oblique regression trees
- Training via the Tree Alternating Optimization (TAO) algorithm
- Rate-distortion performance close to flat (k-means) codebooks
- Much faster encoding due to tree structure and sparsity
- Outperforms previous tree-structured vector quantizers in both distortion and speed

References

[1] L. J. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth, Belmont, Calif., 1984.

[2] M. Á. Carreira-Perpiñán and P. Tavallali. Alternating optimization of decision trees, with application to learning sparse oblique trees. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 1211–1221. MIT Press, Cambridge, MA, 2018.

[3] P. A. Chou, T. Lookabaugh, and R. M. Gray. Optimal pruning with applications to tree-structured source coding and modeling. *IEEE Trans. Image Processing*, 35(2):299–315, Mar. 1989.

[4] S. Dasgupta and Y. Freund. Random projection trees for vector quantization. *IEEE Trans. Information Theory*, 55(7): 3229–3242, July 2009.

[5] Y. Freund, S. Dasgupta, M. Kabra, and N. Verma. Learning the structure of manifolds using random projections. In J. C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 20, pages 473–480. MIT Press, Cambridge, MA, 2008.

[6] R. M. Gray. Vector quantization. *IEEE ASSP Magazine*, 1(2):4–29, Apr. 1984.

[7] J. Lin and J. A. Storer. Design and performance of tree-structured vector quantizers. *Information Processing & Management*, 30(6):851–862, Nov. – Dec. 1994.

[8] L.-M. Po and C.-K. Chan. Adaptive dimensionality reduction techniques for tree-structured vector quantization. *IEEE Trans. Comm.*, 42(6):2246–2257, June 1994.

[9] G. Poggi and R. A. Olshen. Pruned tree-structured vector quantization of medical images with segmentation and improved prediction. *IEEE Trans. Image Processing*, 4(6):734–741, June 1995.

[10] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[11] H. Radha, M. Vetterli, and R. Leonardi. Image compression using binary space partitioning trees. *IEEE Trans. Image Processing*, 5(12):1610–1624, Dec. 1996.