Sparsity in images, a basis better than Fourier basis

Background knowlege: Cooley-Tukey FFT

The *Fourier basis* in $\mathbb{R}^n$ is given by the DFT matrix $(F_n)_{i,j} = \omega^{(i-1)(j-1)}$, where $\omega = e^{-2\pi i/n}$ is the primitive n-th root of unity. F_n is a Vandermonde matrix:

$$F_n = \left(\begin{array}{ccccccccc}
1 & 1 & 1 & \dots & 1 & 1 & \dots & 1 & 1 \\
1 & \omega & \omega^2 & \dots & \omega^{\frac{n}{2}-1} & \omega^{\frac{n}{2}} & \dots & \omega^{n-2} & \omega^{n-1} \\
1 & \omega^2 & \omega^4 & \dots & \omega^{n-2} & \omega^{n} & \dots & \omega^{2n-4} & \omega^{2n-2} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\
1 & \omega^{\frac{n}{2}-1} & \omega^{n-2} & \dots & \omega^{(\frac{n}{2}-1)^2} & \omega^{(\frac{n}{2}-1)\frac{n}{2}} & \dots & \omega^{(\frac{n}{2}-1)(n-2)} & \omega^{(\frac{n}{2}-1)(n-1)} \\
\hline
1 & \omega^{\frac{n}{2}} & \omega^{n} & \dots & \omega^{\frac{n}{2}(\frac{n}{2}-1)} & \omega^{(\frac{n}{2})^2} & \dots & \omega^{\frac{n}{2}(n-2)} & \omega^{\frac{n}{2}(n-1)} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\
1 & \omega^{n-1} & \omega^{2n-2} & \dots & \omega^{(n-1)(\frac{n}{2}-1)} & \omega^{(n-1)\frac{n}{2}} & \dots & \omega^{(n-1)(n-2)} & \omega^{(n-1)^2}
\end{array}\right). \quad (1)$$

The Cooley-Tukey FFT is a divide-and-conquer algorithm for computing the DFT. For simplicity, we assume n is a power of 2, such that we can divide the matrix into 4 blocks:

- The odd columns (blue background), top half:

$$F_{\text{odd, top}} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & \omega^2 & \dots & \omega^{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-2} & \dots & \omega^{(\frac{n}{2}-1)(n-2)} \end{pmatrix} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & (\omega^2) & \dots & (\omega^2)^{\frac{n}{2}-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & (\omega^2)^{\frac{n}{2}-1} & \dots & (\omega^2)^{(\frac{n}{2}-1)(\frac{n}{2}-1)} \end{pmatrix} = F_{\frac{n}{2}} \quad (2)$$

- The even columns (white background), top half:

$$F_{\text{even, top}} = D_{\frac{n}{2}} F_{\frac{n}{2}} \quad (3)$$

  where $D_n = \operatorname{diag}(1, \omega, \omega^2, \dots, \omega^{n-1})$.

- The odd columns (blue background), bottom half:

$$F_{\text{odd, bottom}} = F_{\frac{n}{2}}. \quad (4)$$

  Note $\omega^n = 1$ is ignored.

- The even columns (white background), bottom half:

$$F_{\text{even, bottom}} = -D_{\frac{n}{2}} F_{\frac{n}{2}}, \quad (5)$$

  where the minus sign comes from $\omega^{\frac{n}{2}} = -1$.

Finally, we arrive at the Cooley-Tukey FFT given by:

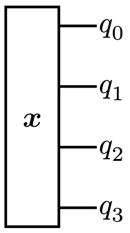
$$F_n \boldsymbol{x} = \begin{pmatrix} I_{\frac{n}{2}} & D_{\frac{n}{2}} \\ I_{\frac{n}{2}} & -D_{\frac{n}{2}} \end{pmatrix} \begin{pmatrix} F_{\frac{n}{2}} & 0 \\ 0 & F_{\frac{n}{2}} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_{\text{odd}} \\ \boldsymbol{x}_{\text{even}} \end{pmatrix} \quad (6)$$

where $\boldsymbol{x}_{\text{odd}}$ and $\boldsymbol{x}_{\text{even}}$ contain the odd and even indexed elements of $\boldsymbol{x}$, respectively. It indicates that the discrete Fourier transormation in $\mathbb{R}^n$ can be decomposed into two smaller discrete Fourier transormations in $\mathbb{R}^{\frac{n}{2}}$ with a diagonal matrix D_n in between. Note applying diagonal matrices can be done in $O(n)$ operations, this decomposition leads to the recurrence relation $T(n) = 2T(\frac{n}{2}) + O(n)$, which solves to $O(n \log n)$ total operations.

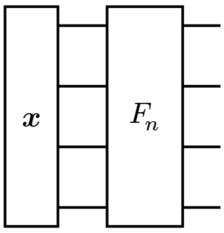
The inverse transformation is given by $F_n^\dagger \boldsymbol{x}/n$. The DFT matrix is unitary up to a scale factor: $F_n F_n^\dagger = nI$.

Tensor network representation of the Cooley-Tukey FFT

This section requires preliminary knowledge of tensor networks (TODO: add reference). In tensor network diagram, a vector of size $n = 2^k$ can be represented as a tensor with k indices, denoting the basis index $i = 2^0 q_0 + 2^1 q_1 + ... + 2^{k-1} q_{k-1}$.

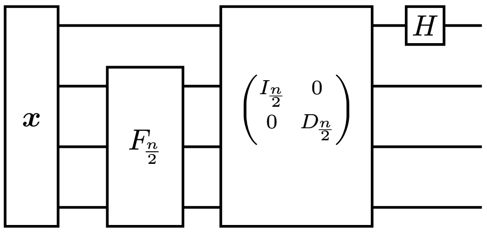


In the following, we aim to find a tensor network decomposition for the linear map F_n:



Step 1: To start, the equation Equation 6 can be represented as the following tensor network:

$$F_n \boldsymbol{x} = \left(\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes I_{\frac{n}{2}} \right) \begin{pmatrix} I_{\frac{n}{2}} & 0 \\ 0 & D_{\frac{n}{2}} \end{pmatrix} \begin{pmatrix} F_{\frac{n}{2}} \boldsymbol{x}_{\text{odd}} \\ F_{\frac{n}{2}} \boldsymbol{x}_{\text{even}} \end{pmatrix}, \qquad (7)$$



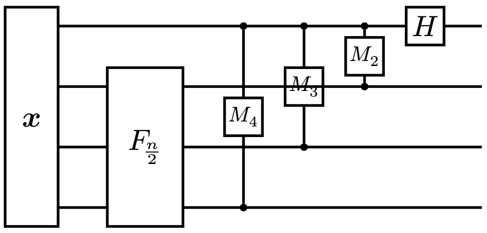
$\boldsymbol{x}$
$F_{\frac{n}{2}}$
$\begin{pmatrix} I_{\frac{n}{2}} & 0 \\ 0 & D_{\frac{n}{2}} \end{pmatrix}$
H

where $H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ is a Hadamard matrix (upto a constant factor).

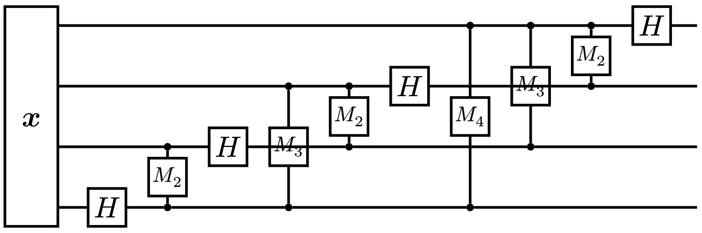
Step 2: Then, we will decompose the diagonal matrix $\begin{pmatrix} I_{\frac{n}{2}} & 0 \\ 0 & D_{\frac{n}{2}} \end{pmatrix}$ into a tensor network. This diagonal matrix corresponds to operation: if q_0 is 0 (odd index), the no operation is applied, otherwise (even index) the operation $D_{\frac{n}{2}}$ is applied. Observe that $D_n = \text{diag}(1, \omega^{\frac{n}{2}}) \otimes \text{diag}(1, \omega, \omega^2, ..., \omega^{\frac{n}{2}-1}) = \text{diag}(1, \omega^{\frac{n}{2}}) \otimes \text{diag}(1, \omega^{\frac{n}{4}}) \otimes ... \otimes \text{diag}(1, \omega)$. We have

$$\begin{pmatrix} I_{\frac{n}{2}} & 0 \\ 0 & D_{\frac{n}{2}} \end{pmatrix} = \text{ctrl}_0\left(\text{diag}(1, \omega^{\frac{n}{4}})_1\right)\text{ctrl}_0\left(\text{diag}(1, \omega^{\frac{n}{8}})_2\right)...\text{ctrl}_0\left(\text{diag}(1, \omega)_{\log_2 n}\right), \qquad (8)$$

where $\text{ctrl}_i(A_j)$ means the target operation applied on A_j is applied only if bit q_i is 1. Here, since the controlled gate is diagonal, it can be represented as a matrix connecting two variables:



In this diagram, $M_k = \begin{pmatrix} 1 & 1 \\ 1 & e^{i\pi/2^{k-1}} \end{pmatrix}$ connects the two qubits involved in the controlled operation, which effectively multilies a phase factor $e^{i\pi/2^{k-1}}$ if two bit are both in state 1. By recursively decomposing the $F_{\frac{n}{2}}$ tensor, we can obtain the following tensor network.

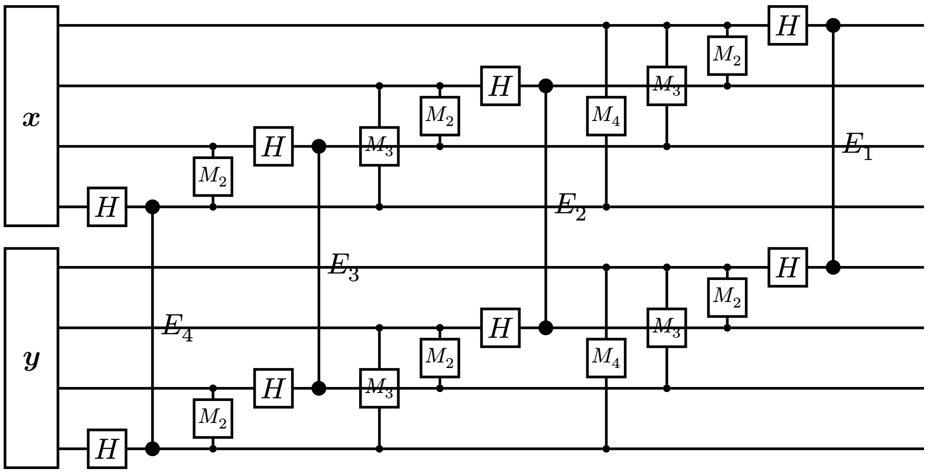


Direct evaluation of this tensor network takes $O(n \log^2 n)$ operations. By respecting the fact that the *controlled phase* operation is a diagonal matrix, we can merge these operations and further reduce the complexity to $O(n \log(n))$.

Entangled Fourier Basis: XY Correlation

In the standard 2D Fourier transform, the x and y coordinates are processed independently. For an image of size $2^n \times 2^n$ (i.e., square images with $m = n$), we apply QFT on the n row qubits and separately on the n column qubits. This independence assumption is often suboptimal for natural images where spatial correlations exist between rows and columns.

We propose an *entangled QFT basis* that introduces controlled-phase gates between x and y qubits after each layer of the QFT circuit. For the square case $m = n$, we use a *one-to-one* entanglement structure where each x qubit x_k is coupled with the corresponding y qubit y_k. This creates correlation between the two spatial dimensions:



The entanglement gates $E_k = \text{diag}(1, 1, 1, e^{i\varphi_k})$ are parameterized controlled-phase gates that couple the k-th qubit from the x-axis with the k-th qubit from the y-axis. For a square $2^n \times 2^n$ image encoded with n row qubits and n column qubits (total $2n$ qubits) and one-to-one coupling between corresponding row/column qubits, we add exactly n entanglement gates, one after each Hadamard layer.

The total transformation becomes:

$$\mathcal{T}_{\text{entangled}} = U_{\text{entangle}} \cdot (F_n \otimes F_n) \tag{9}$$

where $U_{\text{entangle}} = \prod_{k=1}^{n} E_k$ is the product of all entanglement gates, and F_n is the n-qubit QFT applied along each spatial dimension.

Key advantages of this approach:

- Captures diagonal features and cross-dimensional patterns common in natural images
- Maintains $O(n \log n)$ computational complexity in the number n of qubits per spatial dimension (equivalently $O(N \log N)$ for linear image size $N = 2^n$), matching the standard QFT
- Adds exactly n additional real-valued learnable parameters φ_k (one phase per qubit pair), i.e., $O(n)$ in n
- Reduces to standard 2D QFT when all entanglement phases $\varphi_k = 0$

Alternative Basis: Time Evolving Block Decimation (TEBD)

Time Evolving Block Decimation (TEBD) is a tensor network ansatz originally developed for simulating 1D quantum many-body systems. It employs a *brickwork* pattern of nearest-neighbor two-qubit gates applied in alternating layers. For image processing on a $2^n \times 2^n$ grid, TEBD can be adapted by treating the n row qubits and n column qubits as two coupled 1D chains.

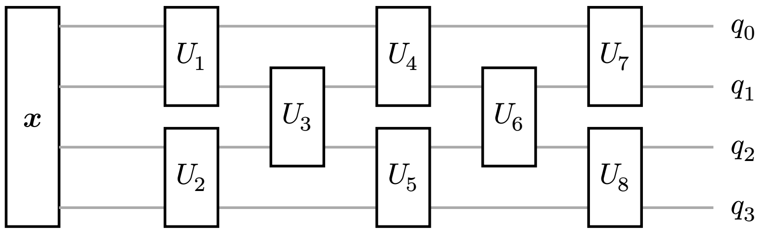


Figure 8: TEBD brickwork circuit for $n = 4$ qubits with $L = 5$ layers: alternating layers of nearest-neighbor two-qubit gates U_k.

In this diagram, each U_k is a parameterized 4×4 unitary matrix acting on two adjacent qubits. Common parameterization choices for U_k include:

1. **Full $U(4)$ unitary**: The most general form with 16 complex parameters constrained by unitarity ($UU^\dagger = I$). This lies on the unitary manifold $U(4)$.

2. **Hardware-efficient ansatz**: Decompose each two-qubit gate as single-qubit rotations followed by an entangling gate:

$$U_k = \left(R_z(\varphi_1)R_y(\theta_1) \otimes R_z(\varphi_2)R_y(\theta_2)\right) \cdot \mathrm{CZ} \cdot \left(R_z(\varphi_3)R_y(\theta_3) \otimes R_z(\varphi_4)R_y(\theta_4)\right) \quad (10)$$

   where $R_y(\theta) = \exp\left(-i\theta\frac{Y}{2}\right)$, $R_z(\varphi) = \exp\left(-i\varphi\frac{Z}{2}\right)$, and CZ is the controlled-Z gate. This uses 8 real parameters per gate.

3. **XX+YY+ZZ interaction**: Inspired by Hamiltonian simulation:

$$U_k = \exp(i(\alpha_k X \otimes X + \beta_k Y \otimes Y + \gamma_k Z \otimes Z)) \quad (11)$$

   with only 3 real parameters $(\alpha_k, \beta_k, \gamma_k)$ controlling the entanglement strength.

Direct evaluation of this tensor network takes $O(nL \cdot 4^2) = O(nL)$ operations for L layers. The parameter space consists of $\lfloor L/2 \rfloor \lfloor n/2 \rfloor + \lceil L/2 \rceil \lfloor (n-1)/2 \rfloor$ two-qubit gates (approximately $(n/2) \cdot L$ gates for large n). The total parameter manifold is:

$$\mathcal{M}_{\mathrm{TEBD}} = \prod_{k=1}^{|\mathrm{gates}|} U(4) \quad (12)$$

For Riemannian optimization, we optimize on this product of unitary manifolds using the same gradient descent approach as the QFT basis.

Alternative Basis: Multi-scale Entanglement Renormalization Ansatz (MERA)

The Multi-scale Entanglement Renormalization Ansatz (MERA) is a hierarchical tensor network that naturally captures *multi-scale correlations*. It consists of alternating layers of *disentanglers* (two-qubit unitaries) and *isometries* (coarse-graining maps), forming a tree-like structure. For $n = 2^k$ qubits, MERA has k layers, with each layer reducing the number of effective qubits by half.

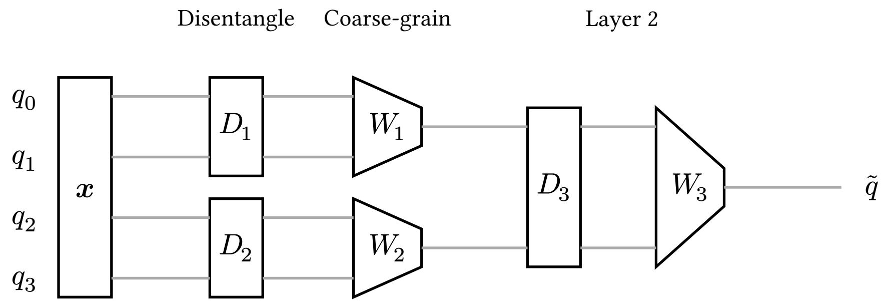


Figure 9: MERA circuit for $n = 4$ qubits: disentanglers D_k (rectangles) remove short-range entanglement, isometries W_k (trapezoids) perform 2-to-1 coarse-graining. Each layer halves the number of effective qubits.

In this diagram, there are two types of parameterized gates:

1. **Disentanglers** $D_k \in U(4)$: Parameterized 4×4 unitary matrices acting on two adjacent qubits before coarse-graining. Same parameterization options as TEBD gates (full $U(4)$, hardware-efficient, or XX+YY+ZZ).
2. **Isometries** $W_k : \mathbb{C}^4 \to \mathbb{C}^2$: Parameterized 2×4 matrices that map two qubits to one qubit (coarse-graining). They satisfy the isometry constraint $W W^\dagger = I_2$, lying on the Stiefel manifold $\mathrm{St}(2, 4)$. Parameterization options include:
   - **Full Stiefel**: Any 2×4 matrix with orthonormal rows, 8 real parameters
   - **Structured**: $W = \begin{pmatrix} \cos\theta & \sin\theta e^{i\varphi_1} & 0 & 0 \\ 0 & 0 & \cos\psi & \sin\psi e^{i\varphi_2} \end{pmatrix}$ with 4 real parameters (block-diagonal structure)

Direct evaluation of this tensor network takes $O(n)$ operations total, since each layer processes $O(2^{k-l})$ qubits at level l and there are $k = \log_2 n$ layers. For $n = 2^k$ input qubits, the parameter count is:

- Layer l: 2^{k-l-1} disentanglers + 2^{k-l-1} isometries
- Total: $\sum_{l=0}^{k-1} 2^{k-l-1} = n - 1$ disentanglers and $n - 1$ isometries

The total parameter manifold is:

$$\mathcal{M}_{\mathrm{MERA}} = \left(\prod_{k=1}^{n-1} U(4)\right) \times \left(\prod_{k=1}^{n-1} \mathrm{St}(2, 4)\right) \tag{13}$$

For Riemannian optimization, we optimize on this product of unitary and Stiefel manifolds. The hierarchical structure naturally captures multi-scale features (edges $\longrightarrow$ textures $\longrightarrow$ objects), with only $O(\log n)$ depth for n qubits.

Learning a better Fourier basis

Observing that in this representation, tensor parameters can be tuned without affecting the computational complexity, e.g. the parameters in M_k and H. Can we find a transformation better than the Fourier basis? Or is Fourier basis already optimal for image processing?

Intuitively, the fourier basis is not optimal for image processing, because:

- the fourier basis assumes periodic boundary condition, which is not suitable for image processing.
- the 2d fourier basis assumes the X and Y coordinates are independent, which is not suitable for image processing.

Tasks

- Create an image dataset $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^{N}$.
- Create a tensor network transformation based on the above QFT circuit, denoted as $\mathcal{T}(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the parameters of the tensor network.
- Variationally optimize the circuit parameters to capture the sparsity of the image. The cost function is

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \|\boldsymbol{x}_i - \mathcal{T}(\boldsymbol{\theta})^{-1}(\text{truncate}(\mathcal{T}(\boldsymbol{\theta})(\boldsymbol{x}_i), k))\|_2^2 \tag{14}$$

  Here, we can choose a different loss function to capture details in the image, e.g. the edges. For simplicity, we use the l_1-norm instead:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \|\mathcal{T}(\boldsymbol{\theta})(\boldsymbol{x}_i)\|_1 \tag{15}$$

  This loss will encourage the tensor network to output a sparse pattern in the “moment space”. It is a standard trick that widely used in *compressed sensing*.
- In the 2D Fourier transformation, the X and Y coordinates are independent. Here we allow X and Y coordinates to correlate with each other in the tensor network basis.
- Add edge detection features.
- Compare the performance with the Fourier basis.