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:

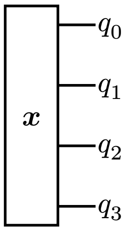
$$F_n \boldsymbol{x} = \begin{pmatrix} I_{\frac{n}{2}} & D_{\frac{n}{2}} \\ I_{\frac{n}{2}} & -D_{\frac{n}{2}} \end{pmatrix} \begin{pmatrix} F_{\frac{n}{2}} & 0 \\ 0 & F_{\frac{n}{2}} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_{\text{odd}} \\ \boldsymbol{x}_{\text{even}} \end{pmatrix} \quad (6)$$

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.

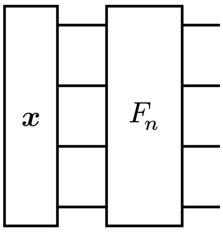
The inverse transformation is given by $F_n^\dagger \boldsymbol{x}/n$. The DFT matrix is unitary up to a scale factor: $F_n F_n^\dagger = nI$.

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 $i = 2^0 q_0 + 2^1 q_1 + \ldots + 2^{k-1} q_{k-1}$.

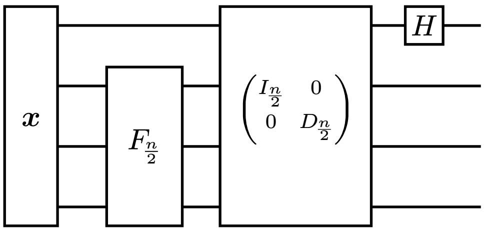


In the following, we aim to find a tensor network decomposition for the linear map F_n:



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

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

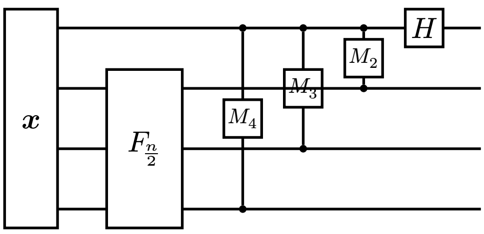


where $H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ is a Hadamard matrix (upto a constant factor).

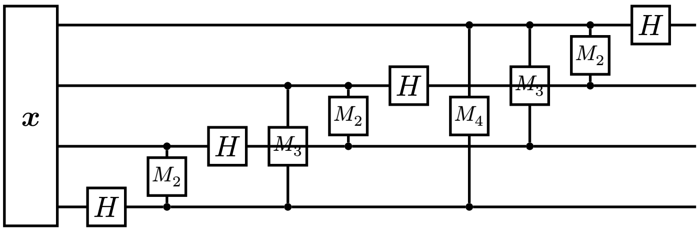
Step 2: Then, we will decompose the diagonal matrix $\begin{pmatrix} I_{\frac{n}{2}} & 0 \\ 0 & D_{\frac{n}{2}} \end{pmatrix}$ into a tensor network. This diagonal matrix corresponds to operation: if q_0 is 0 (odd index), the no operation is applied, otherwise (even index) the operation $D_{\frac{n}{2}}$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

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

where $\operatorname{ctrl}_i\left(A_j\right)$ 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 $O(n \log(n))$.

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

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \|\boldsymbol{x}_i - \mathcal{T}(\boldsymbol{\theta})^{-1}(\mathrm{truncate}(\mathcal{T}(\boldsymbol{\theta})(\boldsymbol{x}_i), k))\|_2^2 \tag{9}$$

  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:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \|\mathcal{T}(\boldsymbol{\theta})(\boldsymbol{x}_i)\|_1 \tag{10}$$

  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.