Estimating the Discrete Fourier Transform using Deep Learning

Jonathan Tuck
Department of Electrical Engineering
Stanford University
jtuck1@stanford.edu

March 16, 2018

Abstract

Throughout a wide variety of fields, the Fourier Transform – and namely the Discrete Fourier Transform (DFT) – is heavily used as a method of analysis of frequency components. In this paper, we show that a simple four-layer neural network with linear activation functions can estimate the DFT to high accuracy. We test our neural network architecture on a variety of signals, and gauge both its accuracy and its speed against the ground truth and its fast, state-of-the-art implementation.

1 Introduction

Throughout a wide variety of fields – from signal processing [OSB99], to medical imaging [Ste00], to optics [Goo96] – the Fourier Transform has been used as an important tool for signal analysis, allowing one to decompose a signal in space or time into its frequency components. Typically, when one wants to analyze a signal, it requires the knowledge of the Fourier Transform (sometimes referred to as the *spectrum*) of the signal. Unfortunately, unknown signals' spectrums are typically only found by computing their Fourier transform and manually inspecting any important attributes, which can be not only a computationally expensive task, but a laborious one as well.

Currently, the method of computing a Fourier transform for a discrete-time signal in $\mathbf{R}^N$, the Discrete Fourier Transform (DFT), can be computed in $O(N^2)$ time. However, there exists a fast and efficient method of computing the DFT, the Fast Fourier Transform (FFT), which can be computed in $O(N \log N)$ time [Bra78, Osg17].

The goal of this project is to estimate the DFT of a bandlimited signal given only its time domain representation. This neural network should be able to estimate the Discrete Fourier Transform of bandlimited signal without having to explicitly compute the Discrete Fourier Transform. Ultimately, the results of this project may, in the future, lead to a deep learning architecture able to compute the DFT of a signal faster than the current state-of-the-art algorithm, the FFT [CT65].

2 The Discrete Fourier Transform

The context and background for this project is abundant, as signal processing has enjoyed a vast treatment in the past century. Although there exist many formulations of the Fourier transform [Osg17], the scope of this project will only deal with the DFT, the version that maps from discrete-time to discrete-time, as that is what computers mathematically can handle. The *N-point DFT* $F : \mathbf{R}^N \to \mathbf{R}^N$ for an N-dimensional discrete-time signal [Bra78, Osg17] $f = (f[0], f[1], \ldots, f[N-1])$ is defined as

$$F\{f\}[m] = \sum_{n=0}^{N-1} f[n] e^{2\pi i m n/N}, \quad m = 0, \ldots, N-1. \tag{1}$$

Figure 1 illustrates a simple example of a discrete-time signal $f[n]$ and the real part of its DFT, $\mathrm{Re}\{F\{f\}[m]\}$.

The DFT in equation (1) can be written as a dense (complex-valued) matrix multiplication, *i.e.*, $F\{f\} = Df$, where $D \in \mathbf{C}^{N \times N}$ is a complex matrix with values $e^{2\pi i m n/N}$ in the (m, n)-index. As the DFT can be computed via a dense matrix multiplication, one can compute the DFT naively in $O(N^2)$ time. In addition, there exists a method to compute the DFT in $O(N \log N)$ time, the FFT. The method by which the FFT computes the DFT is beyond the scope of this project, and we refer the interested reader to [CT65].

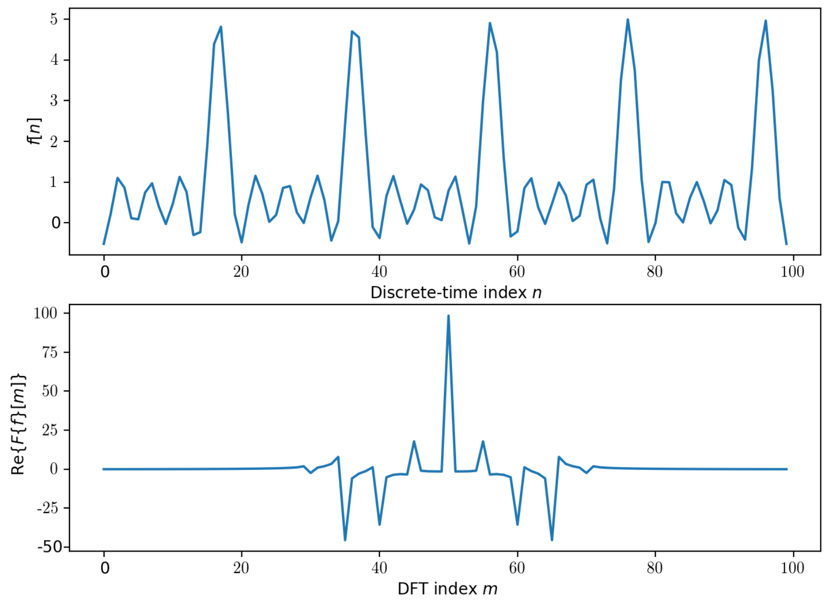

Figure 1: Discrete-time signal $f[n] = \sum_{i=1}^{5} \cos(2\pi i n + i)$ and the real part of its DFT, $\mathrm{Re}\{F\{f\}[m]\}$.

3 Approach

The problem of determining the DFT of a bandlimited signal is a supervised learning problem, and so neural networks for supervised learning problems will be used.

Evaluation metrics. The evaluation of this project shall be based on how close the neural network estimates for a signal's DFT are to the actual signal's DFT. Specifically, the cost function used is

$$\mathcal{J} = (1/m) \sum_{i=1}^{m} \|F\{f_i\} - \hat{F}\{f_i\}\|_2^2,$$

where m is the number of examples in the set, $F\{f_i\} \in \mathbf{R}^N$ is the vector of actual DFT for the i-th example, and $\hat{F}\{f_i\} \in \mathbf{R}^N$ is the DFT estimate for the i-th example. Sometimes, in signal processing literature, this particular cost function is referred to as the *mean squared error* of the estimates [GD10].

3.1 Neural network architecture

The neural network architecture is four layers of fully connected layers, with 100 nodes per layer. Intuitively, as the DFT of a matrix can be described as a matrix multiplication, we pick all of the layers' activation functions to be linear activation functions (and indeed, linear activation functions yielded the greatest performance.) In addition to these hyperparameters selected, we used a learning rate of 0.001, a minibatch size of 200, and a drop-out probability of 1.0 (*i.e.*, no nodes are removed.) We pick these values because they empirically yielded the best training and test performance; that is, our architecture with these hyperparameter values yielded the lowest values of the cost function $\mathcal{J}$ on both the training and test data.

Regularization. Although we attempted to use various forms of regularization (*i.e.*, L2 regularization, L1 regularization, and dropout regularization) we found that overall, our results were better on both the training and test data without any form of regularization.

4 Data

In order to keep the scope of this project manageable, we shall consider only one-dimensional signals (*e.g.*, $f[n] = \cos(n)$), although Fourier transforms and DFTs can be extended to two-dimensional signals (*e.g.*, $g[n, k] = \cos(n)\cos(k)$), three-dimensional signals (*e.g.*, $h[n, k, l] = \cos(n)\cos(k)\cos(l)$), and so on [Osg17]. We shall *a priori* determine a maximum bandwidth, so that aliasing does not occur for our results [OSB99].

As data on computers is inherently discrete, it makes sense to consider discrete-time time series data. Specifically, the time series data is discretized into 100 elements which correspond to evenly spaced indices $t \in [0, 1]$. That is, the input signals are vectors in $\mathbf{R}^{100}$. For the remainder of this paper, when we refer to the DFT, we mean the 100-point DFT.

4.1 Training and test data

Discrete-time time series data is readily abundant and training data can be synthetically generated efficiently. Since all signals are simply sums of sinusoids at different frequencies and amplitudes, it is appropriate to synthetically generate data by simply adding randomly generated sinusoids together, some with various types of noise (*e.g.*, white noise, pink noise, *etc.*.)

For each example, we shall randomly choose the number of sinusoids to be added, and their frequencies. The amplitudes will all be normalized to 1; this is a reasonable normalization to make, as amplitude (to scale) has no effect on the DFT of a signal.

The test data shall be generated in the same way that the training data is generated. In this way, the training data and the test data shall be drawn from the same distributions, so as to minimize variance.

Data splits. In this project, we split our m examples into training, development, and test sets. We have chosen that 90% of the data is dedicated to the training set, and the remaining 10% is dedicated to the test set. We found that we did not need to allocate any data to include a development set.

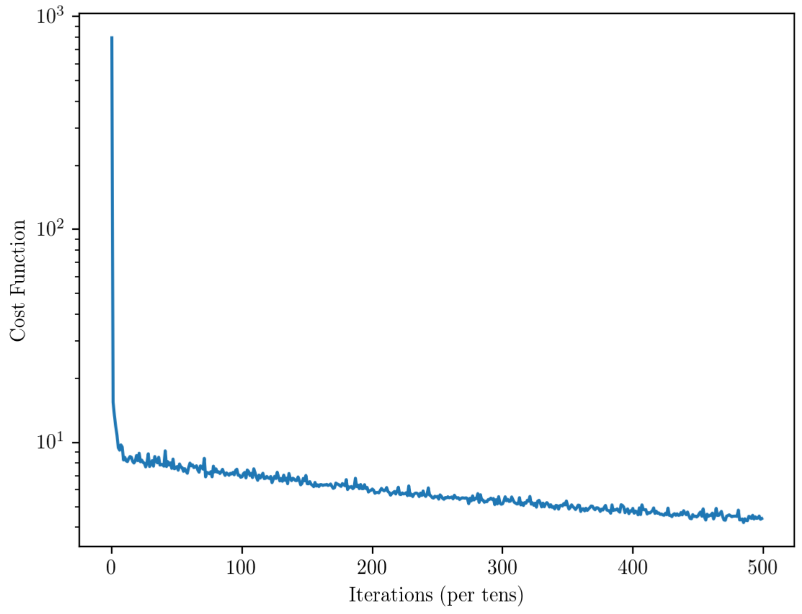

Figure 2: Cost versus epoch on the training data.

5 Results

Figure 2 is a plot of the cost $\mathcal{J}$ versus epoch on the training data for this particular problem instance and initialization.

5.1 Accuracy

For the hyperparameters specified in §3, our neural network architecture achieves an MSE of 2.46×10^{-3} on the training data and an MSE of 6.60×10^{-2} on the test data after 2500 epochs.

Figure 3 is a plot comparing the real part of the DFT of an example in the test set versus its neural network estimate. The two graphs are very similar, but not completely identical; we find that for the particular example in Figure 3, $\|F\{f\} - \hat{F}\{f\}\|_2 / \|F\{f\}\|_2 = 6.55 \times 10^{-3}$. This result suggests that this particular neural network architecutre for estimating the DFT of a vector is valid and can be used successfully.

5.2 Timing

6 Future work

For the field of Fourier analysis with deep learning, there exists a multitude of unanswered questions that stem from this research.

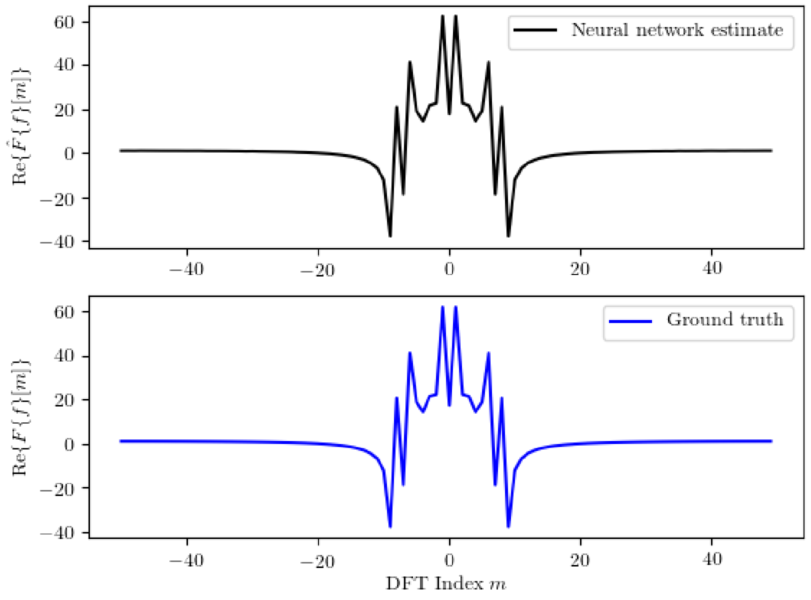

Figure 3: The real part of the DFT of an example (bottom) and the estimate from the output of the neural network.

Exploiting structure. It is well known that many signal processing problems become easier to solve if structure is exploited, such as in compressed sensing, where sparsity is exploited to reduce the minimum sampling rate for perfect signal reconstruction [Don06]. It is possible that these efficients can further be explained or exploited using neural networks with a potentially sparse amount of weights, compared to the neural network of a DFT.

Other transforms. Although the DFT is the most widely known basis transformations, there exists a wide variety of basis transforms that could be exploited using deep learning and with wide-ranging applications, such as the Discrete Cosine transform in image compression [ANR74, YL95], the Radon transform in tomography [Dea07], and the Continuous Wavelet transform [Mal08]. The creation of such a neural network would potentially allow for a much speedier implementation of a particular transform, designed for one particular task (*e.g.*, image reconstruction from a fixed image size.)

7 Contributions and acknowledgements

This project was designed, written, and programmed by Jonathan Tuck. The code for this project can be viewed at `https://github.com/jonathantuck/CS230-project`.

The author would like to thank Xingyu Liu for his insightful comments on how to improve this project, and for suggesting neural network architectures to use for this particular application.

References

[ANR74] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, Jan 1974.

[Bra78] R.N. Bracewell. *The Fourier Transform and its Applications.* McGraw-Hill, Tokyo, second edition, 1978.

[CT65] J. Cooley and J. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[Dea07] S. R. Deans. *The Radon Transform and Some of Its Applications.* Dover Publications, Mineola, N.Y, 2007.

[Don06] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52:1289–1306, 2006.

[GD10] R. Gray and L. Davisson. *An Introduction to Statistical Signal Processing.* Cambridge University Press, New York, NY, USA, 1st edition, 2010.

[Goo96] J.W. Goodman. *Introduction to Fourier Optics.* McGraw-Hill Series in Electrical and Computer Engineering: Communications and Signal Processing. McGraw-Hill, 1996.

[Mal08] S. Mallat. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way.* Academic Press, 3rd edition, 2008.

[OSB99] A. V. Oppenheim, R. W. Schafer, and J. R. Buck. *Discrete-time Signal Processing.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1999.

[Osg17] B. Osgood. *Lectures on the Fourier Transform and its Applications.* McGraw Hill, first edition, 2017.

[Ste00] S. Stergiopoulos. *Advanced Signal Processing Handbook: Theory and Implementation for Radar, Sonar, and Medical Imaging Real-Time Systems.* CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2000.

[YL95] B. Yeo and B. Liu. Volume rendering of DCT-based compressed 3d scalar data. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):29–43, Mar 1995.