Discrete Fourier Transform with Neural Networks

Jabran Akhtar
Norwegian Defence Research Establishment (FFI)
Box 25, 2027 Kjeller, Norway
Email: jabran.akhtar@ffi.no

***Abstract*—The discrete Fourier transform is an important computational tool to retrieve the frequency distribution of a sampled signal. Recent years have also witnessed considerable research activity in neural networks as a mean to solve various signal processing problems. Despite this, it has been an open issue whether a neural network can be trained to return the discrete Fourier transform of its inputs. This paper presents a training methodology for neural networks with non-linear activation functions to replicate and approximate the discrete Fourier transform.**

***Index Terms*—Neural networks, machine learning, Fourier transform, spectral estimation**

I. Introduction

The discrete Fourier transform (DFT) is a well-known technique used to determine the frequency distribution of a finite equally-spaced sampled signal. The DFT has an important role in many wireless applications and transforming a signal or image to frequency domain is often a necessary step for further processing and information extraction [1], [2].

The last couple of years have seen great interest in research and development of neural networks. Deep neural networks have been extensively employed for classification purposes and they have also shown great adaptability to other type of problems [3], [4]. Nevertheless, training an artificial fully-connected neural network to yield a complete discrete Fourier transform has been seen as an intricate task and no clear learning strategy has been proposed. Since the DFT is the output of linear combinations of the inputs, the general understanding has been that a type of neural network with a single layer and linear activation functions can be made to perform this operation [5], [6], [7]. The research activity on this topic has been limited, though it has recently been shown [8] that a convolutional neural network with linear activation functions can be trained to return the magnitude of the DFT. Many neural networks do employ non-linear activation functions and, from a machine learning point of view, it has therefore remained an open problem if such an artificial neural network can be *trained* to converge to the DFT.

In this paper, we propose a supervised learning technique on how a fully-connected neural network can be set up to return the full DFT employing a sigmoid activation function. The short contribution aims to present an easy implementable learning approach to accomplish this. The input data to a neural network is often normalized to avoid scaling issues and speed up the training process. It is our understanding that the standard data normalization approaches such as min-max or z-score have been inadequate for the DFT process as they tend to shrink a signal's dynamic range. We rather make use of maximum absolute normalization which directly leads to the DFT by training on e.g. noisy signals. This also works to demonstrate that the established normalization techniques may not always be most favorable for certain types of problems, particularly those involving transformations.

That a neural network may be able to discover the DFT by itself opens up for several opportunities. For a start, it would allow the computation of the DFT to be carried out via alternative methods employing a traditional sigmoid activation function. Even though a network of a given size may model the DFT exactly, an interesting aspect is the ability to use a smaller network to approximate the full transformation in a novel way. A gradual decline can save computational resources while the essential information needed for further processing may still remain intact [9]. More importantly, a small neural network appropriately designed to be able to construct the DFT on its own, can form part of a bigger deep learning system structure where it itself determines how, where and what type of frequency transformation should occur [10], [11], [12]. Another preeminent application of the technique is in situations when the input data is in time domain and a type of transformed output is expected at the other end. As an example, in [13] gapped slow-time radar data is passed into a trained neural network and the network reconstructs a sparse Doppler profile based on basis pursuit denoising; a process which partly depends on frequency transformation capability. Another application is suggested in [14] where a neural network is trained for super-resolution output in frequency domain.

II. Signal model

In this section we start by describing the basic signal model which is used as the reference point for network training. We assume a complex signal $s[n],\ n = 0, 1, ..., N-1$ where N discrete time domain samples are available. K realizations of $s[n]$ are available, denoted by $s_l[n]$ where $l = 0, 1, ..., K-1$, which are used for the training process. The signals may, optionally, be multiplied element-wise by an arbitrary chosen window function $h[n]$; $h \in \mathbb{R}^N$ and such tapering is often employed in various applications to reduce a signal's amplitude at the beginning and end of the sequence to mitigate spectral leakage. The tapering aspect can thus be incorporated

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/VTC2021-Fall52928.2021.9625247

in the same process, if desired. For each realization of $s_l[n]$, we form a particular normalized version of it, defined as

$$\hat{s}_l[n] = \frac{1}{g_l}(s_l[n] \odot h[n]) \tag{1}$$

where

$$g_l = \max_n \big(|\Re(s_l[n])|, |\Im(s_l[n])| \big), \tag{2}$$

i.e. the maximum absolute value of either the real or imaginary parts of $s_l[n]$. In absolute terms, the largest value of $\hat{s}_l[n]$ will thus be equal to one, while the lowest figure remains undefined. In the above $\odot$ designates element-wise multiplication and it is assumed that $\max_t(h[n]) = 1$ and $g_l > 0$. It turns out that this normalization plays a fundamental role for successful training and operation of the neural network. Other normalization techniques can instead compress the range of the data and hence do not lead to successful outcomes. A discrete Fourier transform is next carried out on the signals:

$$\hat{d}_l[k] = \sum_{n=0}^{N-1} \hat{s}_l[n] e^{-jkn\frac{2\pi}{N}}, \quad k = 0, 1, ..., N-1. \tag{3}$$

Due to the linear properties of the DFT, the resulting transformed data $\hat{d}_l[k]$ can be re-scaled by g_l to attain the absolute values of the recovered solution,

$$d_l[k] = g_l \, \hat{d}_l[k], \tag{4}$$

though this aspect is retained outside the neural network and the training procedure.

By determining the DFT over all K signal realizations, a large training set can be established where each normalized and tapered time domain signal $\hat{s}_l[n]$ is mapped into a Fourier profile $\hat{d}_l[k]$, $\mathbb{C}^N \rightarrow \mathbb{C}^N$. Figure 1 attempts to provide a visual depiction of the training setup. The neural network must be

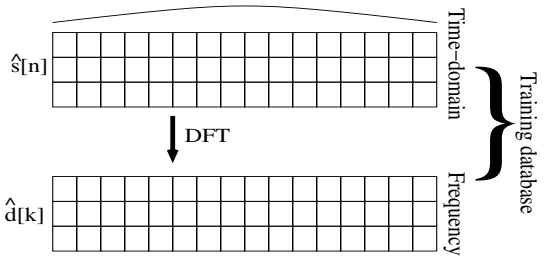

Fig. 1: Training setup

trained to minimize the mean squared error (MSE) between the factual and desired network output. A well-trained network should accordingly be able to evaluate the DFT of any arbitrary signal, as long as the input values are first normalized and re-scaled later. We also point out that the inverse DFT can be computed in a similar manner and hence the same training and/or network design can be applied for that.

III. Neural network design

For the neural network we assume a standard fully-connected feed forwarding structure with an input layer, an output layer with a linear transfer function and one hidden layer with sigmoid activation functions. The input signal and the transformed output data is presumed to be complex valued and we therefore split the data in two, a real and an imaginary part. The $2N$ inputs to the neural networks are normalized, in the manner as described previously, by dividing them by the maximum absolute value of the $2N$ entries. A denormalization is executed at the other end to restore the original scaling. The $2N$ outputs from the final layer are combined at the end to form half as many complex digits. The neural network design is visualized in figure 2.

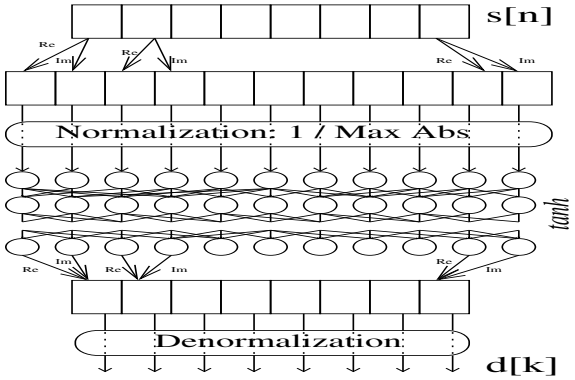

Fig. 2: Neural network process

IV. Training examples and results

A. Training on noisy signals

With the above training setup in mind, a set of simulated signals and their discrete frequency representations were generated to procure an example training set. The objective being to train a neural network to yield, or approximate, a 64-point DFT response. For a direct DFT training we can set

$$s_l[n] \; = \; w_l[n] \tag{5}$$

where $w_l[n]$ is complex white Gaussian noise signal with zero mean and unit variance. A total of $K = 10000$ random noisy signals were generated and the Blackman window was applied before DFT. Noisy signal do not contain any predictable structure and can therefore capture the intricate relationship between complex time domain data and its Fourier transform.

The identical set was used to train a fully-connected feed forwarding network with either 128, 112, 96 or 72 nodes in the one single hidden layer, each node using the hyperbolic tangent sigmoid transfer function. Scale conjugate gradient algorithm was utilized for optimization and the training carried out for up to 500000 epochs with all data in a single learning batch. The convergence was generally quite rapid for the smaller networks and a minima is reached, as can be seen in figure 3, for the various sized networks. The 128 node network continued exhibiting small improvements throughout

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/VTC2021-Fall52928.2021.9625247

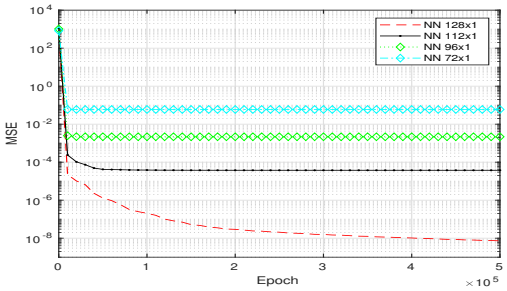

Fig. 3: Training performance

the training session where at the end the MSE materializes at $6.55 \cdot 10^{-9}$. For all practical purposes, the network output was closely resembling the DFT for both real and imaginary values of the training set. The MSE at training end is also given in the second column of table I [1]. Subsequently training, for an initial demonstration, a real valued signal

$$s[n] = 2\cos(0.8\pi n + 2) + 2, \quad n = 1, .., 64, \tag{6}$$

was constructed. The DFT of the signal's magnitude is plotted against the outcomes from the various neural networks in figure 4. The relative error is defined as

$$\eta_S = ||d[k]_{DFT} - d[k]_{NN}||_2 / ||d[k]_{DFT}||_2, \tag{7}$$

where the subscript indicates if the output is from the standard transform (DFT) or a neural network (NN) and given in the third column of table I. The neural network with 128 nodes, i.e. the same number of nodes as the number of inputs and output, yields the exact outcome as of the DFT. That the number of nodes in the hidden layer should be identical to the number of desired output entries to provide an exact outcome has also been observed in [5]. No further improvement is witnessed if the number of nodes in the hidden layer is increased further. On the other hand, the performance from the neural networks declines progressively as the networks hidden layer contain fewer and fewer nodes, resulting in increasingly higher noise floor level. The peaks are nevertheless discriminated well. Smaller networks can be viewed as providing an approximation to the DFT and may still be able to extract information necessary for other learning constructions.

Network size	MSE performance	Test signal relative error, η_S	Test image relative error, η_I
128x1	$6.55 \cdot 10^{-9}$	$3.35 \cdot 10^{-5}$	$4.22 \cdot 10^{-5}$
112x1	$3.75 \cdot 10^{-5}$	0.0033	0.0040
96x1	0.00218	0.0224	0.0253
72x1	0.0593	0.1246	0.1042

TABLE I: Performance error levels

For further evaluation of the generality of the networks trained only on noisy signals, a larger set of signals was generated where a total of five frequencies were incorporated together over $l = 1, ..., 300$ signal realizations. The full

[1]The trained networks are all available for download from: https://doi.org/10.6084/m9.figshare.14980152

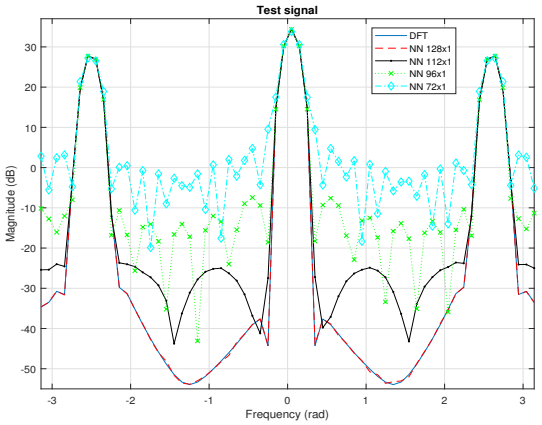

Fig. 4: Test signal outcomes

frequency range was covered for two of the frequency components sweeping from opposite directions while the signal amplitude was set to increment or decrease for the various frequencies as l (x-axis) increased to take account of different scaling. White Gaussian noise was added and the resulting image in the case of standard DFT is provided in figure 5. This can also be seen as a short-time Fourier transform (STFT) executed over the given signals with no overlap.

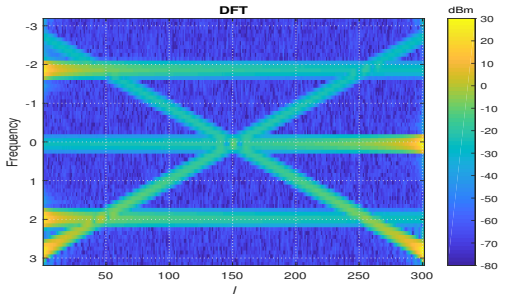

Fig. 5: Standard DFT

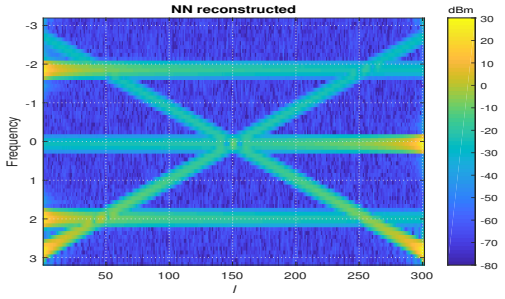

Fig. 6: 128x1 neural network

The comparable results from the trained neural networks are given in figures 6 and 7. To quantify the results, the last column of table I gives numerical error values where the outcomes from various sized networks are compared against the standard DFT. The relative error is now defined as the relative norm

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/VTC2021-Fall52928.2021.9625247

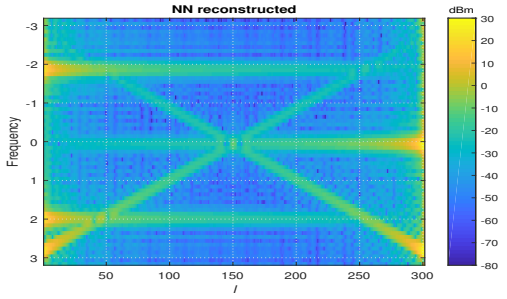

Fig. 7: 72x1 neural network

error between the images formed by the DFT, W_{DFT}, and the neural network image, W_{NN},

$$\eta_I = ||W_{DFT} - W_{NN}||_2 / ||W_{DFT}||_2 \tag{8}$$

In the case of a single hidden layer network with 128 nodes, the approximation to the DFT is again exact, subject to minor numerical errors. When the number of nodes in the hidden layer falls below the number of inputs and outputs, the error rate increases and this is also noticeable in the images as they become more noisy and granular in structure. The major features are nonetheless still approximated and distinguished well.

A neural network with 128 real inputs and 128 output nodes will consist of input and hidden layer matrices of dimensions 128×128 alongside bias vectors of 128 entries. To visualize how the entries are spread, figure 8 presents histogram plots over the distribution of the values in the hidden layer weight matrix and bias vector. The entries of the weight matrix follow a bell shape over a wider interval pointing towards the fact that the network is not simply making linear combinations of its input values. The relative values in the bias vector are typically small which can be attributed to the fact that the input data is normalized to a maximum value of 1.

From a computational point of view, there exist many fast implementations of DFT [1] and a neural network, even with one hidden layer, will not able to achieve the same level of efficiency. A Fast Fourier transform can execute the transformation process in $O(N \log N)$ operations while the complexity of a feed forwarding neural network will be on the order of $O(N^2)$. Nevertheless, in regard to deep networks with hundreds of layers a single additional layer which can potentially compute the DFT can still be of advantage if a Fourier transform type of operation can yield better overall performance and can cut down on other layers.

B. Training on deterministical signals

The previous section demonstrated training on noisy signals but in practice a learning process may have to be carried out on measured data which in many scenarios may only contain real values. With real valued data, the number of input entries to the neural network can be set to exactly N, while the number of nodes in the hidden and output layer should still be at $2N$ for an exact DFT match. To demonstrate training on such a case,

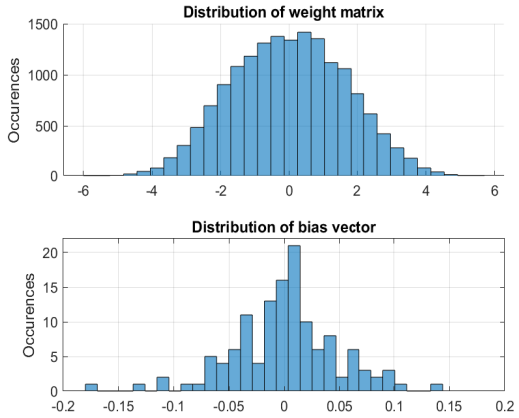

Fig. 8: Weight distribution in the hidden layer

with a sample length of $N = 128$, a clean audio recording of a male voice at 8 kHz was taken advantage of from the freely available NOIZEUS database [15]. The recording contains a total of 22529 samples and random signal blocks of 128 samples from within this were extracted for training. The training was performed with different amounts of data with $K = 160, 192, 256, 1024, 8192$ to evaluate how variation in this impacts the training outcome. Machine learning is a data driven approach and we note that for lower K values, such as $K = 128$, the network is likely to return an exact outcome for the given data set and thus not generalize well. No tapering was employed and the neural network contained one single hidden layer and an output layer each of 256 nodes.

To assess the trained networks, a different audio signal from the same database, corresponding to a crowd of people, with a noise level of 15.1 dB was put to use. The zoomed in standard DFT spectrogram of the evaluation signal can be seen in figure 9 (top) with an overlap of 50% (64 samples). The middle plot demonstrates the neural network result when trained with $K = 160$ while the bottom represents the outcomes with $K = 1024$. The symmetrical property of DFT for real signals can be observed and even with $K = 160$ the spectrum is visually very close to the original DFT though small deviations can be observed. Table II displays numerical errors with outcomes from various sized data levels as compared against the standard DFT, over the full sample duration of about three seconds. The errors are low, however, as expected, more available training data results in better outcomes.

K, number of training signals	MSE performance	Spectrogram image, relative error η_I
160	$1.21 \cdot 10^{-6}$	0.3209
192	$1.99 \cdot 10^{-6}$	0.1272
256	$1.08 \cdot 10^{-6}$	0.0017
1024	$1.34 \cdot 10^{-6}$	0.0003
8192	$1.61 \cdot 10^{-6}$	0.0001

TABLE II: Performance error levels

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/VTC2021-Fall52928.2021.9625247

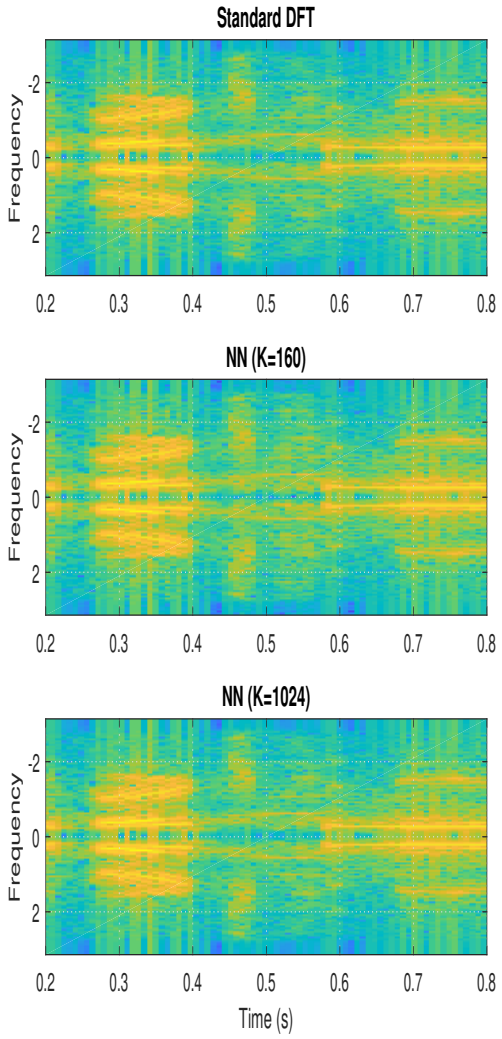

Fig. 9: Spectrogram with DFT and NN

V. Conclusion

This paper proposed a training strategy for neural networks in order to return the discrete Fourier transform. The key aspect being training over an alternative normalized version of signals. The results show that with an appropriate structure one can indeed train a fully-connected feed forwarding neural network with one single hidden layer to learn to yield the complex DFT which can also optionally incorporate a windowing function. Smaller networks can be used as an approximation to the transform. The use of neural network in this context opens up for the possibilities where only time domain data is fed into networks and the network itself determines if a transformation is desirable for solving a given task.

References

[1] A. V. Oppenheim and R. W. Schafer, *"Discrete-Time Signal Processing"*. Prentice-Hall, 1989.

[2] W. K. Pratt, *"Digital Image Processing"*. Wiley & sons Inc., 1991.

[3] H. Huttunen, *"Deep neural networks: A signal processing perspective"*. Handbook of Signal Processing Systems (Third Edition), S. S. Bhattacharyya, E. F. Deprettere, R. Leupers, and J. Takala, Eds. Springer, 2019.

[4] D. Gündüz, P. de Kerret, N. Sidiropoulos, D. Gesbert, C. Murthy, and M. van der Schaar, "Machine learning in the air," *IEEE J. Sel. Areas Comm.*, vol. 37, no. 10, pp. 2184–2199, Oct. 2019.

[5] R. Velik, "Discrete Fourier transform computation using neural networks," in *International Conference on Computational Intelligence and Security*, 2008, pp. 120–123.

[6] M. Namba and Y. Ishida, "Pitch synchronous Fourier transform using neural networks," in *International Conference on Neural Networks*, 1995.

[7] O. Moreira-Tamayo and J. P. D. Gyvez, "Filtering and spectral processing of 1-D signals using cellular neural networks," in *IEEE International Symposium on Circuits and Systems*, 1996, pp. 76–79.

[8] M. A. García and A. E. Destéfanis, "Spectrogram prediction with neural networks," in *XXIV Congreso Argentino de Ciencias de la Computación*, 2018, pp. 42–51.

[9] J. M. Winograd and S. H. Nawab, "Incremental refinement of DFT and STFT approximations," *IEEE Signal Processing Letters*, vol. 2, no. 2, pp. 25–27, Feb. 1995.

[10] M. Kulin, T. Kazaz, I. Moerman, and E. D. Poorter, "End-to-end learning from spectrum data: A deep learning approach for wireless signal identification in spectrum monitoring applications," *IEEE Access*, vol. 6, pp. 18 484–18 501, March 2018.

[11] M. Crisan, *"Dynamic Neural Network Model of Speech Perception" in "Advances in Intelligent Systems and Computing" S. Bhatia, S. Tiwari, K. Mishra, M. Trivedi (Eds.)*. Singapore: Springer, 2019.

[12] Y.-H. Pan, C.-H. Lin, and T.-S. Lee, "GAN-CRT: A novel range-Doppler estimation method in automotive radar systems," in *Proc. IEEE VTC Spring*, 2020.

[13] J. Akhtar, "Sparse range-Doppler image construction with neural networks," in *Proc. of IEEE Radar Conference*, 2020, pp. 291–296.

[14] ——, "Augmenting radar Doppler resolution with neural networks," in *European Signal Processing Conference*, 2021.

[15] Y. Hu and P. Loizou, "Subjective evaluation and comparison of speech enhancement algorithms," in *Speech Communication*, 2007, pp. 588–601.

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/VTC2021-Fall52928.2021.9625247