2017 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 25—28, 2017, TOKYO, JAPAN

LEARNING APPROXIMATE NEURAL ESTIMATORS FOR WIRELESS CHANNEL STATE INFORMATION

Tim O'Shea, Kiran Karra, and T. Charles Clancy

Virginia Tech
Department of Electrical and Computer Engineering, Arlington, VA
oshea@vt.edu, kiran.karra@vt.edu, tcc@vt.edu

ABSTRACT

Estimation is a critical component of synchronization in wireless and signal processing systems. There is a rich body of work on estimator derivation, optimization, and statistical characterization from analytic system models which are used pervasively today. We explore an alternative approach to building estimators which relies principally on approximate regression using large datasets and large computationally efficient artificial neural network models capable of learning non-linear function mappings which provide compact and accurate estimates. For single carrier PSK modulation, we explore the accuracy and computational complexity of such estimators compared with the current gold-standard analytically derived alternatives. We compare performance in various wireless operating conditions and consider the trade offs between the two different classes of systems. Our results show the learned estimators can provide improvements in areas such as short-time estimation and estimation under non-trivial real world channel conditions such as fading or other non-linear hardware or propagation effects.

1. INTRODUCTION

Estimation is a critical component of many signal processing systems. We focus in this paper on several key synchronization tasks which are used widely in the reception of single carrier modulated phase-shift keying (PSK) signals. Specifically, we look at carrier frequency offset (CFO) estimation and timing estimation which are used to recover the time and frequency of a received signal transmission to enable demodulation and information recovery. Excellent estimators exist for these tasks and are derived analytically from expressions for the signal model and the channel model using techniques such as maximum-likelihood (MLE) or minimum-mean squared error (MMSE) optimization metrics to produce an expression for an estimator. This approach works well, and has worked in many communications systems over the year; however it suffers from several shortcomings:

1. It relies on an accurate analytic model of the signal and channel, the signal is man-made and typically easy to model, but hardware defects and distortion effects along with channel propagation effects or distributions in specific environments are rarely captured in great detail when performing this optimization. Analytic estimators derived under Gaussian channel assumptions are often used by default.
2. It requires a manual analysis process to derive a well conditioned estimator for a specific signal type, modulation type, and/or set of reference tones which is time consuming and often leads to time and cost in system engineering.
3. Algorithms formed through the analytic process are typically kept compact (minimum number of terms) in order to facilitate the ease of analysis. However, from a power consumption perspective, such a compact form may not be the most concurrent form of an estimator, where algorithmic concurrency often leads to the lowest power algorithm on mobile platforms.

Artificial neural networks (ANNs) have been used for regression tasks previously [1], [2], many of these involving signal processing tasks or transformations. However numerous advances have been made in ANNs over the past few years including significant improvements in gradient descent, regularization, network architecture and activation functions, as well as the use of many-core compute platforms such as graphics processing units (GPUs) to realize and train very large networks rapidly. These advances have led to the growth of 'deep learning', where training very large neural networks which were previously in-feasible (for instance in the 90s when many of the prior works in this area were published) can now be accomplished easily with commercial hardware and open source software such as TensorFlow [10] and Keras [9]. Recent efforts [11] have shown this class of deep learning for physical layer representations and algorithms to be effective and competitive with modern baselines.

The outline of our paper is as follows. We begin by discussing expert estimator approaches to estimating the timing

978-1-5090-6341-3/17/$31.00

and frequency offsets for PSK burst signals. We then describe a new, deep learning based approach for estimating these quantities. After estimation is discussed, we conduct experiments with both of these approaches and compare the performance. Finally, concluding remarks are provided.

2. BACKGROUND - EXPERT ESTIMATION

In this section, we introduce and describe the expert features based center frequency offset estimation and timing offset estimation techniques.

2.1. Center Frequency Offset Estimation

A common approach to center frequency offset estimation is to use an FFT based technique which estimates the frequency offset by using a periodogram of the m^{th} power of the received signal [4]. The frequency offset detected by this technique is then given by (1)

$$\Delta\hat{f} = \frac{F_s}{N \cdot m} \underset{f}{\operatorname{argmax}} \left| \sum_{k=0}^{N-1} r^m[k] e^{-j2\pi kt/N} \right| \qquad (1)$$
$$\left(-\frac{R_{sym}}{2} \leq f \leq \frac{R_{sym}}{2} \right),$$

where m is the modulation order, $r(k)$ is the received sequence, R_{sym} is the symbol rate, F_s is the sampling frequency, and N is the number of samples. The algorithm searches for a frequency that maximizes the time average of the m^{th} power of the received signal over various frequencies in the range of $\left(-\frac{R_{sym}}{2} \leq f \leq \frac{R_{sym}}{2} \right)$. Due to the algorithm operating in the frequency domain, the center frequency offset manifests as the maximum peak in the spectrum of $r^m(k)$. Fig. 1 shows an example cyclic spectrum for a QPSK signal with a 2500 Hz center frequency offset, where the peak indicates the center frequency offset for the burst.

2.2. Timing Offset Estimation

In traditional wireless communications systems, timing offset estimation is performed by matched filtering the received sequence to a known preamble sequence. The time-offset which maximizes the output of the matched filter's convolution is then taken to be the time-offset of the received signal. Matched filtering can be represented by (2)

$$y(k) = \sum_{k=-\infty}^{k=\infty} h[n-k] r[k], \qquad (2)$$

where $h[k]$ is the preamble sequence. The matched-filter is known as the optimal filter for maximizing the signal to noise ratio (SNR) in the presence of additive stochastic white noise.

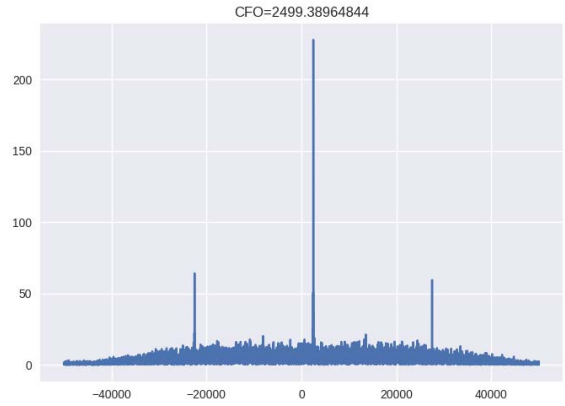

Fig. 1. CFO Expert Estimator Power Spectrum with simulated 2500 Hz offset

Table 1. Common Regression Loss Functions

Method	Expression
MSE	$L_{MSE}(y,\hat{y}) = \sum_i (y_i - \hat{y}_i)^2$
MAE	$L_{MAE}(y,\hat{y}) = \sum_i abs\,(y_i - \hat{y}_i)$
log-cosh	$L_{LogCosh}(y,\hat{y}) = \sum_i log\,(cosh\,(y_i - \hat{y}_i))$
Huber	$L_{Huber}(y,\hat{y}) = \sum_i \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2 & abs(y_i - \hat{y}_i) < 1 \\ (y_i - \hat{y}_i) & abs(y_i - \hat{y}_i) \geq 1 \end{cases}$

3. TECHNICAL APPROACH FOR LEARNED ALGORITHMS

In this section, we introduce our approach for learning algorithms to estimate the CFO and timing offsets.

Our approach to learned estimator generation relies on construction, training and evaluating an ANN based on a representative dataset. When relying on learned estimators, much of work and difficulty lies in generating a dataset which accurately reflects the final usage conditions desired for the estimator. In our case, we produce numerous examples of wireless emissions in complex baseband sampling with rich channel impairment effects which are designed to match the intended real world conditions the system will operate in. We associate target labels from ground truth for center frequency offset and timing error which are used to optimize the estimator.

To train an ANN model, we consider the minimization of mean-squared error (MSE) and log-cosine hyperbolic (log-cosh) [6] and Huber loss functions (shown in table 1). The latter are known to have improved properties in robust learning, which may benefit such a regression learning task on some datasets and tasks. In our initial experiments in this paper, we observe the best quantitative performance using the MSE loss function which we shall use for the remainder.

We search over a large range of model architectures using

Table 2. ANN Architecture Used for CFO Estimation

Layer	Output dimensions
Input	(nsamp,2)
Conv1D + ReLU	(variable,32)
AveragePooling1D	(variable,32)
Conv1D + ReLU	(variable,128)
Conv1D + ReLU	(variable,256)
Linear	1

Table 3. ANN Architecture Used for Timing Estimation

Layer	Output dimensions
Input	(2048,2)
Conv1D + ReLU	(511,32)
Conv1D + ReLU	(126,64)
Conv1D + ReLU	(30,128)
Conv1D + ReLU	(2,256)
Dense + Linear	(1)

Adam [8] to perform gradient descent to optimize for model parameters based on our training dataset. This is done by computing a loss function (e.g. L_{MSE}) and updating the weights of the neural network model iteratively using back-propagation of loss gradients.

This model search and optimization process ideally seeks a model of minimal computational complexity which achieves a satisfactory level of performance (the frontier of efficient models represents a trade-off between model complexity and accuracy). However in this paper we do not limit or optimize for model complexity, choosing only the best performing model based on accuracy (lowest MSE) alone.

The ANN architectures used for our performance evaluation are shown below, both are stacked convolutional neural networks with narrowing dimensions which map noisy high dimensional raw time series data down to a compact single valued regression output. In the case of CFO estimation architecture shown in Table 2, we find that an average pooling layer works well to help improve performance and generalization of the initial layer feature maps, while in the timing estimation architecture in table 3 no-pooling, or max-pooling tends to work better. This makes sense on an intuitive level as CFO is distilling all symbols received throughout the input into a best frequency estimate, while timing in a traditional matched filter sense, is derived typically from a maximum response at a single offset.

Primitives operations described in the architecture tables include 1D convolutional layers (Conv1D) [3], the rectified linear unit activations (ReLU) [5], max/average pooling layers [7], and fully connected (Dense) layers with no non-linearity applied (Linear).

4. PERFORMANCE ANALYSIS

In this section, we discuss the methodology used to evaluate expert estimators with learned estimators. We begin by discussing how the datasets used for evaluating the performance between these approaches are generated. We then discuss the test methodology. Finally, we discuss the results of the different approaches.

4.1. Data Generation

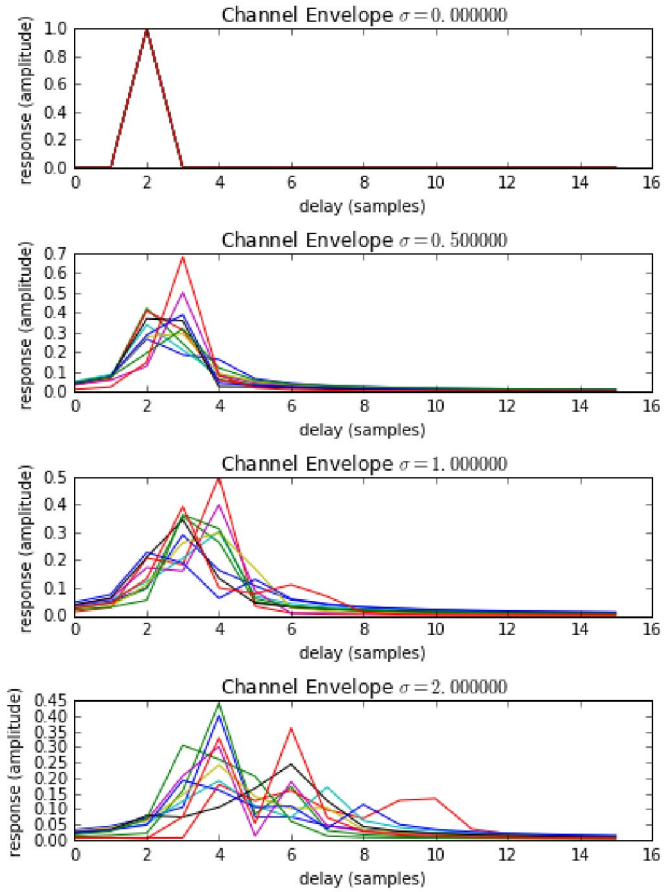

Fig. 2. Impulse Response Modes Compared

We generate two different sets of data for evaluating the performance of the two competing approaches. All generated data are based off of QPSK bursts with equiprobable independent and identically distributed (IID) symbols, and shaped with a square root raised cosine (RRC) filter with a roll-off $\beta = 0.25$ and a filter span of 6, and sampled at 400 kHz with a symbol rate of 100 kHz. We consider 4 channel conditions, additive white Gaussian noise (AWGN) with no fading, and three cases of Rayleigh fading with varying mean delay spreads in samples of $\sigma = 0.5, 1, 2$. Amplitude envelopes for a number of complex valued channel responses for each of

these delay spreads are shown in figure 2 to provide some visual insight into the impact of Rayleigh fading effects at each of these delays. For the last case, significant inter-symbol interference (ISI) is present in the data.

The first dataset generated is the timing dataset, in which we prepended the burst with a known preamble of 64 symbols and random noise samples at the same SNR as the data portion of the burst. The number of noise samples preprended is drawn from a $\mathcal{U} \sim (0, 1.25)$, in units of milliseconds. Additionally, a random phase offset drawn from a $\mathcal{U} \sim (0, 2\pi)$ is introduced for each burst in the dataset.

The second dataset generated is the center frequency offset data, in which every example burst has a center frequency offset drawn from a $\mathcal{U} \sim (-50e3, 50e3)$ distribution, in units of Hz. The bounds of this correspond to half the symbol rate, $R_{sym}/2$. Additionally, a random phase offset drawn from a $\mathcal{U} \sim (0, 2\pi)$ is introduced for each burst in the dataset.

These datasets are generated for SNR's of 0 dB, 5 dB, and 10 dB and for an AWGN channel and three different Rayleigh fading channels with different mean delay spread values (0.5, 1, and 2) representing different levels of reflection in a given wireless channel environment. We store the label of the timing offset and center frequency offsets as ground truth for training and evaluation.

4.2. Test Methodology and Experimental Results

For each dataset generated above we optimize network weights using Adam [8] for 100 epochs, reducing the initial learning rate of $1e-3$ by a factor of two for each 10 epochs with no reduction in validation loss, ultimately using the parameters corresponding to the epoch with the lowest validation loss. With the datasets generated above, we then compute the test error using a separate data partition between ground truth labels for timing and center frequency offset and predicted values generated using both expert and deep learning/ANN based estimators. The mean absolute error (MAE) of the estimator is used as our metric for comparison.

4.2.1. Timing Results

In the timing estimation comparison, we show estimator MAE results in figure 3, for each model AWGN(τ, χ) and Fading(τ, χ) where τ is the mean delay spread, and χ is the SNR. Inspecting these results we can see that the traditional matched filtering maximum likelihood approach (MF/MLE) achieves excellent performance under the AWGN channel condition (awgn channel model). We can see significant degradation of the MF/MLE baseline accuracy under the fading channel models however as a simple matched filter MLE timing estimation approach has no ability to compensate for the expected range of channel delay spreads. In this case the artificial neural network / machine learning (ML/ANN) estimator approach on average can not attain equivalent performance in all or even most cases. However, we see that this approach does attain a MAE r within the same order of magnitude, and does in some fading cases achieve a lower MAE in the case of a fading channel.

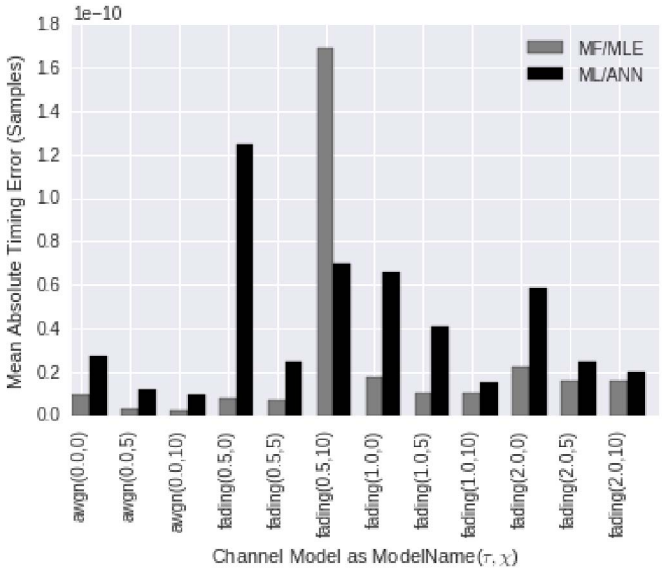

Fig. 3. Timing Estimation MAE Comparison

4.2.2. Center Frequency Results

Quantitative results for estimation of center frequency offset error are shown in figures 4,5,6,7, summarizing the performance of both the baseline maximum likelihood (MLE) method with dashed lines and the ML/ANN method with solid lines. We compare the mean absolute center frequency estimate error for each method at a range of different estimator block input length sizes. As moment based methods generally improve for longer block sizes, we compare performance over a range of short-time examples to longer-time examples.

In the AWGN case, in figure 4 we can see that for 5 and 10dB SNR cases, by the time we reach a block size of 1024 samples, the baseline estimator is doing quite well, and for larger block sizes (above 1024 samples) with SNR of at least 5dB, performance of the baseline method is generally better. However, even in the AWGN case, for small block sizes we are able to achieve lower error using the ML/ANN approach, even at low SNR levels of near 0dB.

In the cases of fading channels shown in figures 5,6,7, we can see that performance of the baseline estimator degrades enormously from the AWGN case under which it was derived when delay spread is introduced. Performance gets perpetually worse as σ increases from 0.5 to 2 samples of mean delay spread. In the case of the ML/ANN estimator, we also see a degradation of estimator accuracy as delay spread increases, but the effect is not nearly as dramatic, ranging from 3.4 to

23254 Hz in the MLE case (almost a 7000x increase in error) versus a range of 2027 to 3305 Hz in the ML/ANN case (around a 1.6x increase in error).

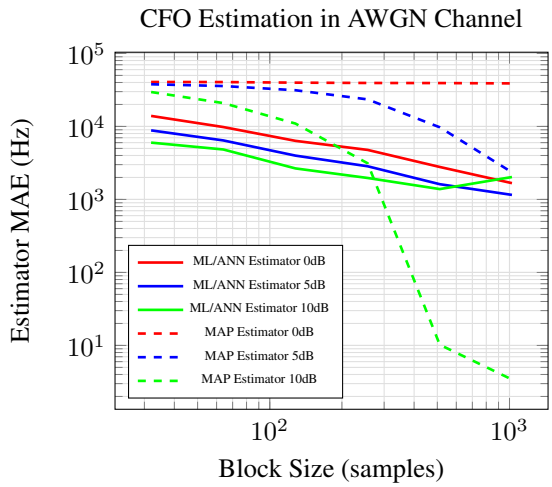

Fig. 4. Mean CFO Estimation Absolute Error for AWGN Channel

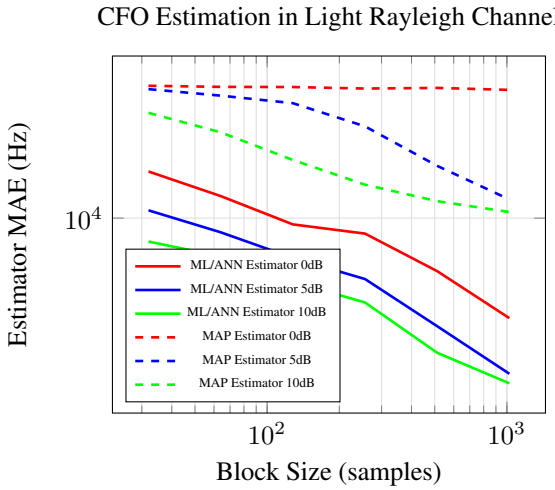

Fig. 5. Mean CFO Estimation Absolute Error (Fading σ=0.5)

4.3. Computational Complexity

In this section, we discuss the computational complexity of the expert estimators and the neural network based estimators. Table 5 compares the approximate number of floating point operations (FLOPs) required to compute the center frequency offset estimate for both the expert estimator described above, and the neural network based estimator. The FLOP counts for the expert estimator were derived by using the FFTW estimate for the number of flops required to compute an FFT, which scales with the the FFT size, N, as $5Nlog_2(N)$. The

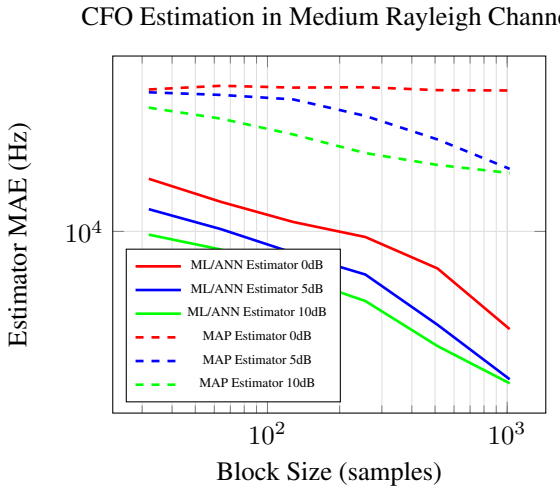

Fig. 6. Mean CFO Estimation Absolute Error (Fading σ=1)

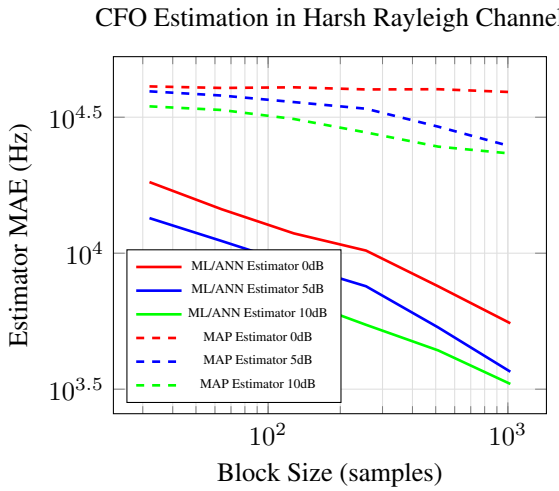

Fig. 7. Mean CFO Estimation Absolute Error (Fading σ=2)

FLOP counts reported are a function of the desired CFO frequency estimation resolution (1 Hz), the sampling frequency (400 kHz), and the input sequence length (parameterized in the table). It is noticed in Table 5 that the expert estimator FLOP counts do not change with an increasing input sequence length. This is because the FFT size required is, in this case, dominated by the desired frequency resolution of 1 Hz, rather than the input sequence length in this particular scenario.

The complexity of each layer of the neural network may be computed by considering the number of adds and multiplies within each layer. These are given in terms of number of filter length L, input channels ch_i, output channels ch_o, output width K, input size N_i and output size N_o and pool size p. Table 4 enumerates the complexity of each layer type in terms of multiplies and adds as a function of layer parameters.

Table 4. Approx. Layer Complexity

Layer/Op	Expression
Conv1D/Mul	$L * ch_i * ch_o * K$
Conv1D/Add	$L * (ch_i + 1) * ch_o * K$
Dense/Mul	$N_i * N_o$
Dense/Add	$(N_i + 1) * N_o$
AvgPool/Add	$N_o * p$

Table 5. CFO FLOP Count

Sample Size	Expert Estimator (MFlop)	NN Estimator (MFlop)
32	5.374	3.01
64	5.374	2.89
128	5.374	4.36
256	5.374	6.92
512	5.374	12.59
1024	5.374	23.71

In the case of the ANN complexity measurement, we observe that for small networks we are able to obtain lower complexity (rather than taking a high-resolution FFT). While for the larger example sizes we perform a worst case of 5-10x more operations per estimation. These results are not at all bad considering the ANN architectures were not optimized to minimize complexity. When observing the complexity per layer for instance, it is clear that certain layers (for instance with large filter sizes) dominate the operation count, where they could likely be easily alleviated with smaller filter sizes and to obtain similar performance. Effects of architecture optimization and reduced data representation precision will likely yield a complexity reduction on the order of 10-100x from these floating point numbers, providing an extremely competitive low-complexity estimator approximation with the baseline.

Similarly, Table 6 shows the approximate number of floating point operations required to estimate the timing offset by both the expert estimators and the neural network based estimators. The bulk of the FLOPs required for the estimation using the expert estimator depend on the input sequence length, because a matched filter must correlate across the entire sequence to find the optimal starting point.

5. CONCLUSION

Estimation has been a fundamental building block of signal processing and wireless communications systems since their inception. Optimization of analytically derived and statistically well-formed estimators has long been the building block upon which such systems have always been built, but is a non-trivially difficult process under channel and signal models of high complexity. Our results are mixed, showing that in some cases existing benchmark estimators such as the matched filter are quite hard to beat using learned methods, but results in the same order of magnitude can be obtained readily given sufficient good labeled data and limited signal knowledge (for instance the ANN timing estimator had no knowledge of the preamble used, the modulation type, the pulse shaping filter, etc) which potentially offers a lower implementation complexity if the cost of obtaining good labeled data is lower than that of imparting all the known reference signal information into the expert estimator.

In center frequency estimation, we see that under ideal channel condition under which many commonly used estimators are derived, performance is very good for large block size and high SNR cases, offering a level of precision which we are currently unable to attain from similar learned ANN approaches. However, we can see that at lower SNR levels, for smaller block sizes, and for harsh non-impulsive fading channel conditions the learned ANN based estimator approach appears to offer significant potential for improvement.

The widespread use of learned approximate estimators is still in its infancy, such systems have not yet gained widespread acceptance or proven themselves to demonstrate all desirable properties under varying conditions, but we believe what we have shown here provides a valuable step towards this goal demonstrating that under certain constraints and requirements such systems do appear to outperform their baseline equivalents significantly based solely on regression of labeled datasets.

From a complexity standpoint we have shown that the ANN estimator complexity is within the same order of magnitude as the traditional estimators, is lower in some cases, and holds the potential to be significantly lower in complexity given additional optimization.

Going forward, approximate estimation based on regression of large datasets holds an extremely promising avenue of research when building practical engineering systems. As we have shown in this paper, there are conditions such as short-time, and low-SNR conditions where accuracy gains can be achieved against current baselines, and there are also conditions where complexity reduction can be attained to reduce the computational complexity, size, weight, and power required in order to obtain a comparable estimate. All of these properties must be traded in an engineering system design when selecting a design approach. In future work we seek to further reduce the computational complexity through additional architecture optimization and further characterize the conditions under which each approach holds significant benefits or drawbacks versus the other.

Table 6. Timing FLOP Count

Sample Size	Expert Estimator (MFlop)	NN Estimator (MFlop)
1024	1.05165	9.35

References

[1] D. F. Specht, "A general regression neural network," *IEEE transactions on neural networks*, vol. 2, no. 6, pp. 568–576, 1991.

[2] A Cochocki and R. Unbehauen, *Neural networks for optimization and signal processing*. John Wiley & Sons, Inc., 1993.

[3] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[4] Y. Wang, K. Shi, and E. Serpedin, "Non-data-aided feedforward carrier frequency offset estimators for qam constellations: A nonlinear least-squares approach," *EURASIP Journal on Advances in Signal Processing*, vol. 2004, no. 13, p. 856 139, 2004.

[5] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[6] O. Catoni *et al.*, "Challenging the empirical mean and empirical variance: A deviation study," in *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, Institut Henri Poincaré, vol. 48, 2012, pp. 1148–1185.

[7] M. Hoai12, "Regularized max pooling for image categorization," 2014.

[8] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[9] F. Chollet, *Keras*, 2015.

[10] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[11] T. J. O'Shea and J. Hoydis, "An introduction to machine learning communications systems," *CoRR*, vol. abs/1702.00832, 2017. [Online]. Available: `http://arxiv.org/abs/1702.00832`.