A. APPENDIX FOR "DENOISING NEURAL NETWORKS FOR MAGNETIC RESONANCE SPECTROSCOPY"

Authors: Natalie Klein* Amber J. Day*† Harris Mason* Michael W. Malone* Sinead A. Williamson†
*Los Alamos National Laboratory, Los Alamos, NM 87545, USA
†Department of Statistics and Data Sciences, University of Texas at Austin, Austin, TX 78712, USA

A.1. Dataset details

When generating synthetic data based on Equation 1, we use a time vector with 1,024 points at a sample spacing of 1.8×10^{-5} seconds to correspond to the experimental data. We draw parameters T_2 and σ independently according to a Uniform($1 \times 10^{-3}, 1 \times 10^{-2}$) distribution. Phase shift ϕ is drawn according to a Uniform($-\pi, \pi$) distribution. The noise is drawn from one of two distributions: white Gaussian noise ('white') and experimental noise measurements from a nuclear quadrupole resonance (NQR) instrument with no relevant signal present ('measured'). The measured noise consists of 4,096 independent measurements of noise at the same sampling rate with duration 16,384 samples; we first split the independent measurements into training, validation, and test sets before dividing into pieces of length 1,024 to ensure no single measurement vector was reused in the validation or test sets. We consider signals centered at frequency $f \sim 0$Hz (to emulate complex-demodulated data) and signals centered at a higher frequency $f \sim F$, with $F = 1600$Hz (chosen to have signal power in a frequency band for which the measured noise has elevated power). Given a choice of f, the frequency of an individual signal in the data set is drawn as Uniform($f-150, f+150$). We control the signal-to-noise ratio (SNR) of the signals by varying A relative to standardized noise (variance equal to one), with a low-SNR regime, $A \sim$ Uniform$(0.1, 1.0)$, and a high-SNR regime, $A \sim$ Uniform$(2.1, 3.0)$. The experimental data comes from nuclear magnetic resonance measurements of $NaNO_2$. Complex demodulation at the expected resonance frequency was performed during data collection, resulting in complex-valued time series data expected to be centered at $f \sim 0$Hz. It consists of the same time vector as the synthetically generated signal and upon inspection was seen to have high SNR, with visible exponential decay trend in the noisy signals.

A.2. SNR across datasets

Signal-to-noise ratio (SNR) is defined as the ratio of the power of a signal (P_{signal}) to the power of background noise (P_{noise}), SNR $= \frac{P_{signal}}{P_{noise}}$. We reference the logarithmic decibel scale of the SNR (SNR$_{dB}$) for ease of comparison: SNR$_{dB}$ $= 10\log_{10}\left(\frac{P_{signal}}{P_{noise}}\right) = 10\log_{10}(\text{SNR})$. Since the SNR$_{dB}$ values are approximately the same across noise distributions and between the in-distribution and out-of-distribution datasets, we have only reported the values for the white-noise and in-distribution datasets.

SNR Type	Frequency	Minimum SNR$_{dB}$	Average SNR$_{dB}$	Maximum SNR$_{dB}$
low	$f \sim 0$	-35.6	-17.7	-7.3
low	$f \sim 1600$	-35.2	-17.0	-7.2
high	$f \sim 0$	-10.0	-3.1	2.5
high	$f \sim 1600$	-9.7	-2.4	2.6

Table A3: Observed minimum, average, and maximum SNR (dB) in the synthetic datasets.

A.3. Complex- and real-valued NN approaches

In the main text, we focused on CV-NNs compared to one RV-NN approach working on concatenated real and imaginary components. For the fully-connected AE, the real and imaginary components were directly concatenated into a single vector; for the convolutional CTN, the real and imaginary components were treated as two channels. This approach allows the NN to learn relationships between the real and imaginary components of the signal. We also investigated two other RV-NN approaches (Fig. A4), but found the performance was consistently worse. In particular, we considered one RV-NN applied independently to the real and imaginary components, or separate RV-NNs applied independently to the real and imaginary components. Neither of these approaches considers relationships between the real and imaginary components, which could explain the poorer performance.

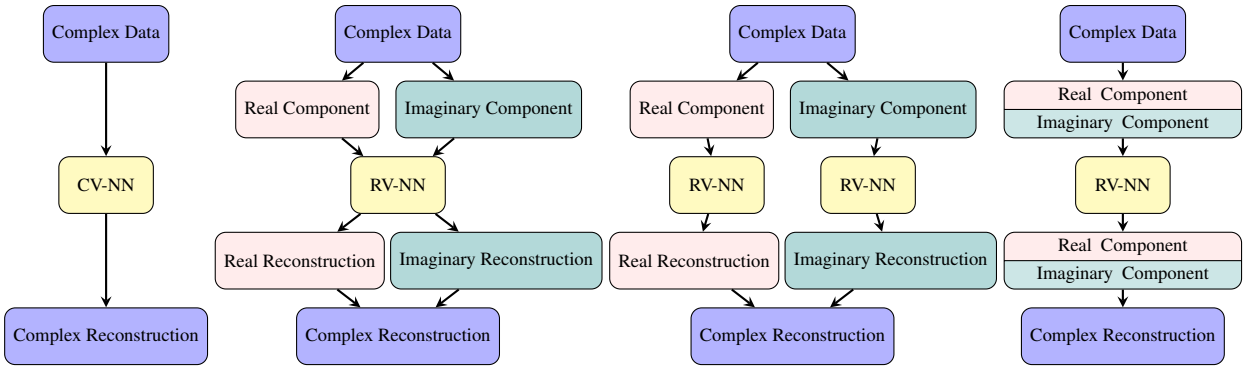


Fig. A4: CV-NN and RV-NN architectures. From left to right, ComplexNet, DualReal1, DualReal2, DualReal1C. ComplexNet operates directly on complex data with complex-valued activation and loss functions. DualReal1 splits data into real and imaginary but processes them with a single real-valued network. DualReal2 splits into real and imaginary and processes them with two separate real-valued networks. DualReal1C concatenates the real and imaginary components into a single real-valued vector (or two-channel vector) and then processes them with a single real-valued network which allows the model to learn unconstrained relationships between real and imaginary components. The other models are either unable to learn these relationships at all (DualReal1 and DualReal2) or have some symmetries imposed on the relationships (ComplexNet).

A.4. Autoencoder implementation details

Four different undercomplete autoencoder architectures were implemented (Fig. A4). Each of these four architectures were trained in the following way. The full 1024 dimensional space was mapped down to 10 dimensions, then 5 dimensions, before being mapped back to 10 dimensions and finally the full 1024 dimensions. Training was performed using the Adam optimizer. Each epoch trained on the full data set. The maximum number of training epochs was 1000 with early stopping after 5 consecutive increases in validation loss. The loss was calculated as denoising loss, meaning the squared loss was calculated between the reproduction from the model and the clean version of the signal. A learning rate of 0.01 was used. The complex-valued and real-valued MSE loss function and ReLU activation function were implemented since they had the best overall performance when tested against complex ReLU, ReLU, complex cardioid, complex phase TanH, Hardtanh for activation functions and complex MSE, MSE, complex logMSE, logMSE, complex sdr, L1Loss, SNRloss for loss functions.

A.5. ConvTasNet implementation details

We based our implementation on the `torchaudio` implementation of ConvTasNet but with two important modifications: first, we used a deep rather than single layer encoder/decoder, and second, we developed an adaptation to use complex-valued convolutions and activation functions. We set the depth of the encoder and decoder to three layers. The key variation we explored in deviating from default `torchaudio` hyperparameter settings was changing the window size to 32, 64, or 128 samples. For the complex-valued CTN, we used complex convolutions and complex PReLU activation functions to mirror the real-valued CTN [12]. Due to GPU memory constraints, we used minibatch training with a batch size of 256 signals. We optimized the networks for a maximum of 50 epochs with the Adam optimizer (learning rate: 3×10^{-3}), keeping the model with the highest validation error as a form of early stopping. We found that using the log of the squared reconstruction error resulted in more stable training for CTN; this loss function is similar to an SNR-based loss.

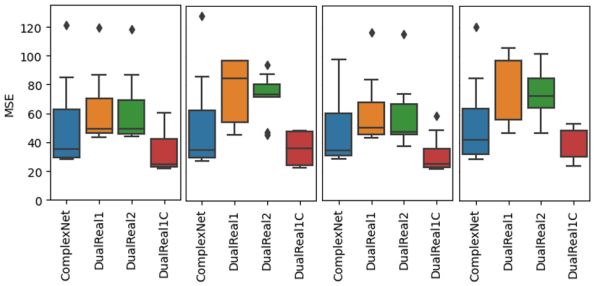
A.6. Baseline denoising method details

The time-domain deconvolution method uses `least_squares` from the Python `scipy.optimize` package to fit the generating signal model to the noisy data. We used knowledge of the data generating distribution to set the scale and initial values of the parameters to be optimized, so that the method would have in a sense the best performance possible, but we note that in practice, the procedure can be very sensitive to these generally unknown values. The SSA method was implemented using the Fast Gradient Cadzow approach [20]; it works by creating a lagged time series matrix from each signal, then performing SVD, and iteratively thresholding to determine which components correspond to signal. The wavelets baseline method processes the amplitude of the raw NQR signal using the Python package PyWavelets with sym4 wavelet, 12 decompositions, Bayes denoising, and soft thresholding [8].

A.7. Complex- and real-valued autoencoder comparisons

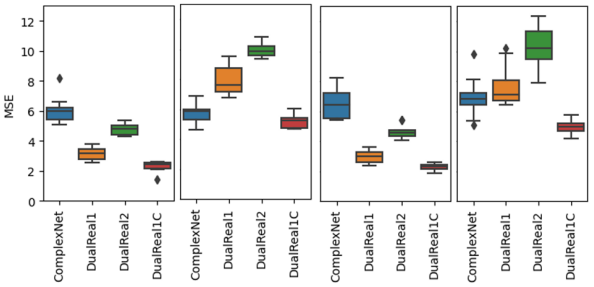
Comparison of the performance of four different autoencoders, ComplexNet, DualReal1, DualReal2, and DualReal1C across different random initialization providing insight into each model's sensitivity to initialization. It can be seen that DualReal1C generally has the lowest variability across seeds and the lowest MSE. We believe it performs best due to the density of its layers, as it has the highest number of weights, and because of the flexibility in learning the relationships between the real and imaginary components of the data. While ComplexNet is over-all outperformed by DualReal1C, more often than not, it outperforms the more naïve RV-NNs, DualReal1, and DualReal2. We believe this is due to its ability to learn symmetrically constrained relationships between the real and imaginary components of the data, which DualReal1 and DualReal2 lack. Other things made apparent by A4 is that the sensitivity to initialization is much higher for all autoencoders across the low-SNR data sets and is much lower across the high-SNR data sets.

	White noise				Measured noise			
	$f \sim 0$		$f \sim F$		$f \sim 0$		$f \sim F$	
Model	Mean	SD	Mean	SD	Mean	SD	Mean	SD
ComplexNet	50.6	29.8	51.4	31.7	48.1	23.6	53.3	28.4
DualReal1	61.8	23.5	77.2	21.3	60.9	22.3	81.4	22.9
DualReal2	61.4	23.3	72.7	14.7	58.0	22.2	74.2	17.6
DualReal1C	33.0	13.0	36.1	10.8	32.1	12.1	41.9	11.5



(a) low SNR

	White noise				Measured noise			
	$f \sim 0$		$f \sim F$		$f \sim 0$		$f \sim F$	
Model	Mean	SD	Mean	SD	Mean	SD	Mean	SD
ComplexNet	6.03	0.87	5.80	0.67	6.51	0.96	6.94	1.28
DualReal1	3.16	0.44	7.95	0.92	2.99	0.42	7.70	1.28
DualReal2	4.78	0.37	9.96	0.48	4.65	0.44	10.4	1.34
DualReal1C	4.96	0.46	2.30	0.34	5.19	0.43	2.30	0.23



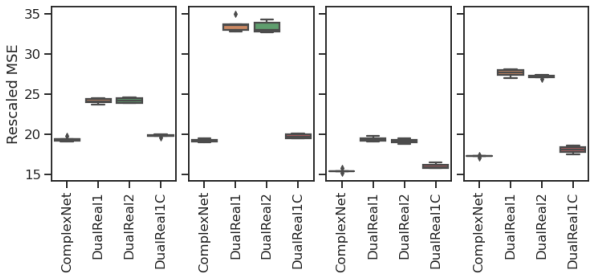
(b) high SNR

Table A4: Comparing mean $\times 10^{-3}$ and standard deviation (SD $\times 10^{-3}$) of MSE from four different autoencoders.

A.8. Complex- and real-valued ConvTasNet comparisons

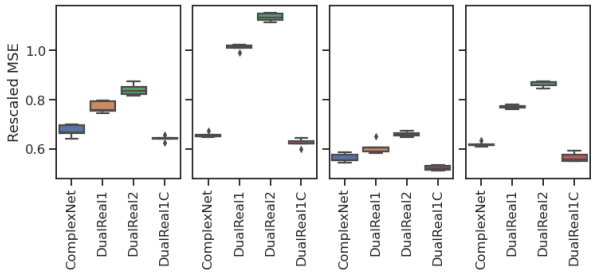
For a fixed window size (128; results similar for other window sizes), we compare the different complex- and real-valued ConvTasNet models across different seeds. Table A5 shows the mean and SD of the rescaled MSE of each model in the ensemble for the different complex- and real-valued networks across different training regimes; boxplots below the tables show the distributions of rescaled MSE across each ensemble. In low SNR, the complex-valued CTN appears to outperform the DualReal1C CTN with smaller ensemble variation; the other real-valued approaches (DualReal1 and DualReal2) perform significantly worse. In high SNR, the DualReal1C model appears to perform slightly better than the complex CTN.

	White noise				Measured noise			
	$f \sim 0$		$f \sim F$		$f \sim 0$		$f \sim F$	
Model	Mean	SD	Mean	SD	Mean	SD	Mean	SD
ComplexNet	19.38	0.23	19.24	0.16	15.43	0.21	17.25	0.10
DualReal1	24.15	0.28	33.59	0.74	19.34	0.24	27.62	0.38
DualReal2	24.19	0.30	33.29	0.65	19.14	0.23	27.18	0.18
DualReal1C	19.80	0.13	19.74	0.27	16.04	0.30	18.07	0.40



(a) low SNR

	White noise				Measured noise			
	$f \sim 0$		$f \sim F$		$f \sim 0$		$f \sim F$	
Model	Mean	SD	Mean	SD	Mean	SD	Mean	SD
ComplexNet	0.67	0.02	0.66	0.01	0.56	0.02	0.62	0.01
DualReal1	0.77	0.02	1.01	0.01	0.61	0.02	0.77	0.01
DualReal2	0.84	0.02	1.13	0.01	0.66	0.01	0.86	0.01
DualReal1C	0.64	0.01	0.62	0.01	0.52	0.01	0.57	0.02



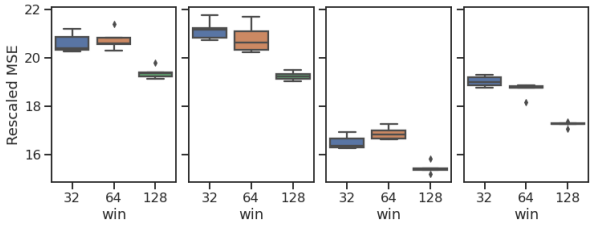
(b) high SNR

Table A5: Comparing mean $\times 10^{-3}$ and standard deviation (SD $\times 10^{-3}$) of MSE from four different ConvTasNet.

A.9. ConvTasNet window selection

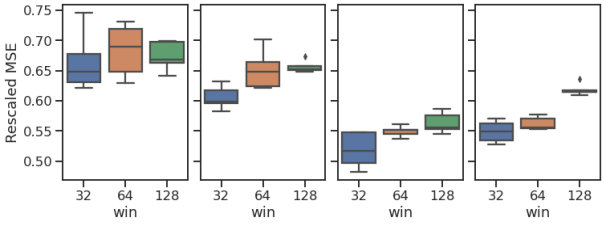
Based on previous work, we determined that varying the window size in ConvTasNet would be the manipulation most likely impact model performance, so we trained models with window lengths of 32, 64, and 128 (in number of samples). Because Table 1 suggests that the complex-valued ConvTasNet was better overall than the real-valued ConvTasNet across all window sizes, we here investigate how window size impacted the performance of the complex-valued ConvTasNet. Table A6a shows results for low SNR, where the table gives the mean and standard deviation of the rescaled test set MSE across the ensemble of models, with corresponding boxplots below the table showing the distribution of rescaled MSE values for each window length (across the ensemble members). In this case, it appears that the longest window length performed best across all training data sets. Interestingly, the opposite appeared to be true for high SNR (Table A6b), which suggests that smaller window sizes performed better, though the differences between window sizes were less pronounced than low SNR. These results suggest that in high SNR, the window length is less impactful on the model performance, while in low SNR, the longer window length tends to provide better performance, perhaps because more total signal and noise information is contained in each window.

	White noise				Measured noise			
	$f \sim 0$		$f \sim F$		$f \sim 0$		$f \sim F$	
Model	Mean	SD	Mean	SD	Mean	SD	Mean	SD
32	20.60	0.36	21.15	0.36	16.50	0.26	19.02	0.20
64	20.74	0.37	20.79	0.54	16.87	0.23	18.67	0.26
128	19.38	0.23	19.24	0.16	15.43	0.21	17.25	0.10



(a) low SNR

	White noise				Measured noise			
	$f \sim 0$		$f \sim F$		$f \sim 0$		$f \sim F$	
Model	Mean	SD	Mean	SD	Mean	SD	Mean	SD
32	0.67	0.04	0.61	0.02	0.52	0.03	0.55	0.02
64	0.68	0.04	0.65	0.03	0.55	0.01	0.56	0.01
128	0.67	0.02	0.66	0.01	0.56	0.02	0.62	0.01



(b) high SNR

Table A6: Comparing mean $\times 10^{-3}$ and standard deviation (SD $\times 10^{-3}$) of MSE from different complex ConvTasNet window sizes.

A.10. Out-of-distribution data and results

We generated out-of-distribution (OOD) test sets in two ways. First, we applied trained neural networks (NNs) to test sets that had different noise distributions (trained on white noise, applied to data with measured noise, or vice versa) with all other data characteristics held fixed. To measure performance, we calculated a standardized mean squared error (MSE), in which each true signal and prediction was first rescaled by the true signal amplitude A. As a measure of ensemble uncertainty, we calculated the standard deviation (SD) in the individual model MSE values. Table A7 shows that the NN performance in terms of ensemble

mean squared error (MSE) remains stable when the noise distribution changes, and that the ensemble standard deviation (SD) also remains stable. These results suggest that the networks are robust to some changes in the noise distribution at deployment.

SNR	f	w → w	w → m	m → m	m → w
high	0	0.54(0.04)	0.44(0.03)	0.44(0.03)	0.56(0.03)
high	F	0.50(0.02)	0.48(0.02)	0.45(0.02)	0.51(0.02)
low	0	17.9(0.23)	14.6(0.19)	14.1(0.21)	18.3(0.28)
low	F	17.8(0.16)	17.1(0.26)	15.8(0.10)	18.2(0.17)

(a) Ensemble MSE(SD) $\times 10^{-3}$ for CTN (CV) models.

SNR	f	w → w	w → m	m → m	m → w
high	0	1.86(0.25)	1.81(0.16)	1.89(0.13)	1.95(0.12)
high	F	4.36(0.22)	4.28(0.23)	4.10(0.35)	4.17(0.34)
low	0	22.5(5.68)	22.3(0.06)	21.8(0.87)	22.4(8.44)
low	F	23.6(4.83)	23.4(5.28)	29.3(6.34)	29.5(5.75)

(b) Ensemble MSE(SD) $\times 10^{-3}$ for AE (DualReal1C) models.

Table A7: Ensemble MSE(SD) $\times 10^{-3}$ for NN models trained under four SNR/central frequency conditions (rows); columns indicate training and test noise distributions, where 'w' is white noise and 'm' is measured noise. Here, w → w indicates that the model was trained with white noise and applied to test data with white noise, while w → m indicates the model was trained with white noise and applied to test data with measured noise (OOD), and similar for the remaining two columns. Generally, moving from in-distribution to OOD test sets does not increase MSE or ensemble uncertainty, indicating robustness to noise distribution.

Second, we generated test sets that had different signal frequency distributions ($f \sim$ 300Hz or $f \sim$ 1900Hz) with the frequency of an individual example drawn as Uniform($f-150, f+150$). This means that during test time the signal frequencies are shifted so that they do not overlap the frequencies seen during training, but are still relatively close to the training frequencies. Table A8 shows that performance degrades substantially in this case. In high SNR, the CTN ensemble SD increases, suggesting that the ensemble uncertainty could help indicate OOD data, but this pattern does not hold in low SNR. This result suggests that training sets should be designed to contain the range of frequencies anticipated during deployment.

	White noise				Measured noise			
Trained	$f \sim 0$		$f \sim 1600$		$f \sim 0$		$f \sim 1600$	
Tested	$f \sim 0$	$f \sim 300$	$f \sim 1600$	$f \sim 1900$	$f \sim 0$	$f \sim 300$	$f \sim 1600$	$f \sim 1900$
AE	1.86(0.25)	29.5(0.32)	4.36(0.22)	35.0(0.18)	1.89(0.13)	29.7(0.09)	4.10(0.35)	34.7(0.15)
CTN	0.54(0.04)	16.3(1.26)	0.50(0.02)	21.5(3.44)	0.44(0.03)	14.7(1.42)	0.45(0.02)	18.1(4.09)

(a) Relative MSE for out of distribution high-SNR data.

	White noise				Measured noise			
Trained	$f \sim 0$		$f \sim 1600$		$f \sim 0$		$f \sim 1600$	
Tested	$f \sim 0$	$f \sim 300$	$f \sim 1600$	$f \sim 1900$	$f \sim 0$	$f \sim 300$	$f \sim 1600$	$f \sim 1900$
AE	22.5(5.68)	47.1(1.41)	23.6(4.83)	48.6(1.13)	21.8(0.87)	47.1(1.32)	29.3(6.34)	48.4(1.16)
CTN	17.9(0.23)	65.2(0.24)	17.8(0.16)	65.9(0.45)	14.1(0.21)	59.5(0.26)	15.8(0.10)	63.2(0.67)

(b) Relative MSE for out of distribution low-SNR data.

Table A8: Ensemble MSE(SD) $\times 10^{-3}$ for NN models trained with central frequencies near 0 and 1600Hz applied to test data at 0 or 1600Hz (in-distribution) or 300 or 1900Hz (OOD). Moving from in-distribution to OOD signals results in higher MSE across both NN models. In high SNR, the ensemble uncertainty increases when moving to OOD data, suggesting it could be used as an indicator of OOD data during deployment. However, the same pattern does not hold for low SNR data.

6. REFERENCES

[1] Joel B Miller, "Nuclear quadrupole resonance detection of explosives," in *Counterterrorist Detection Techniques of Explosives*, pp. 157–198. Elsevier, 2007.

[2] Allen N Garroway, Michael L Buess, Joel B Miller, Bryan H Suits, Andrew D Hibbs, Geoffrey A Barrall, Robert Matthews, and Lowell J Burnett, "Remote sensing by nuclear quadrupole resonance," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 39, no. 6, pp. 1108–1118, 2001.

[3] Alexander A Koukoulas and MA Whitehead, "Observations in nuclear quadrupole resonance frequency temperature dependence," *Chemical Physics Letters*, vol. 167, no. 5, pp. 379–382, 1990.

[4] K Ramani, S Ganapathy, and R Srinivasan, "A Fourier transform approach to Voigt profile analysis and its application to nuclear magnetic resonance," *Journal of Magnetic Resonance (1969)*, vol. 24, no. 2, pp. 231–237, 1976.

[5] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion.," *Journal of Machine Learning Research*, vol. 11, no. 12, 2010.

[6] Yi Luo and Nima Mesgarani, "Conv-TasNet: Surpassing ideal time–frequency magnitude masking for speech separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 8, pp. 1256–1266, 2019.

[7] Cristian Monea and Nicu Bizon, *Signal Processing and Analysis Techniques for Nuclear Quadrupole Resonance Spectroscopy*, Springer Nature, 2021.

[8] Qi Shan, "EMD and wavelet methods for interference cancellation in quadrupole resonance detection," *Journal of Research in Science and Engineering (JRSE)*, vol. 4, no. 5, pp. 149–158, 2022.

[9] Samuel D Somasundaram, Andreas Jakobsson, and Naveed R Butt, "Countering radio frequency interference in single-sensor quadrupole resonance," *IEEE Geoscience and Remote Sensing Letters*, vol. 6, no. 1, pp. 62–66, 2008.

[10] K Thulasiram Varma, Rangababu Peesapati, Ch V Rama Rao, AK Rajarajan, Makarand Dixit, and Gopal Joshi, "A novel SSA-NLLSF approach for denoising NQR signals," *Applied Magnetic Resonance*, vol. 52, no. 11, pp. 1601–1613, 2021.

[11] Cristian Monea, "Enhancing deep learning nuclear quadrupole resonance detection using transfer learning and autoencoders," *Expert Systems with Applications*, vol. 207, pp. 118093, 2022.

[12] C Trabelsi, O Bilaniuk, Y Zhang, D Serdyuk, S Subramanian, JF Santos, S Mehri, N Rostamzadeh, Y Bengio, and CJ Pal, "Deep complex networks," *arXiv preprint arXiv:1705.09792*, 2017.

[13] Karn N Watcharasupat, Thi Ngoc Tho Nguyen, Woon-Seng Gan, Shengkui Zhao, and Bin Ma, "End-to-end complex-valued multidilated convolutional neural network for joint acoustic echo cancellation and noise suppression," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 656–660.

[14] Akira Hirose and Shotaro Yoshida, "Generalization characteristics of complex-valued feedforward neural networks in relation to signal coherence," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 4, pp. 541–551, 2012.

[15] Natalie Klein and Amber Day, "SplitML," `https://github.com/lanl/SplitML`, 2022, Accessed: 2022-10-19.

[16] Yao-Yuan Yang, Moto Hira, Zhaoheng Ni, Artyom Astafurov, Caroline Chen, Christian Puhrsch, David Pollack, Dmitriy Genzel, Donny Greenberg, Edward Z Yang, et al., "Torchaudio: Building blocks for audio and speech processing," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 6982–6986.

[17] Vinith Kishore, Nitya Tiwari, and Periyasamy Paramasivam, "Improved speech enhancement using TCN with multiple encoder-decoder layers," in *Interspeech*, 2020, pp. 4531–4535.

[18] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[19] David S Broomhead and Gregory P King, "Extracting qualitative dynamics from experimental data," *Physica D: Nonlinear Phenomena*, vol. 20, no. 2-3, pp. 217–236, 1986.

[20] Haifeng Wang, Jian-Feng Cai, Tianming Wang, and Ke Wei, "Fast Cadzow's algorithm and a gradient variant," *Journal of Scientific Computing*, vol. 88, no. 2, pp. 1–21, 2021.