On the appropriateness of complex-valued neural networks for speech enhancement

Lukas Drude[1], *Bhiksha Raj*[2], *Reinhold Haeb-Umbach*[1]

[1]Department of Communications Engineering – University of Paderborn
[2]School of Computer Science – Carnegie Mellon University
drude@nt.upb.de, bhiksha@cs.cmu.edu, haeb@nt.upb.de

Abstract

Although complex-valued neural networks (CVNNs) – networks which can operate with complex arithmetic – have been around for a while, they have not been given reconsideration since the breakthrough of deep network architectures. This paper presents a critical assessment whether the novel tool set of deep neural networks (DNNs) should be extended to complex-valued arithmetic. Indeed, with DNNs making inroads in speech enhancement tasks, the use of complex-valued input data, specifically the short-time Fourier transform coefficients, is an obvious consideration. In particular when it comes to performing tasks that heavily rely on phase information, such as acoustic beamforming, complex-valued algorithms are omnipresent. In this contribution we recapitulate backpropagation in CVNNs, develop complex-valued network elements, such as the split-rectified non-linearity, and compare real- and complex-valued networks on a beamforming task. We find that CVNNs hardly provide a performance gain and conclude that the effort of developing the complex-valued counterparts of the building blocks of modern deep or recurrent neural networks can hardly be justified.

Index Terms: complex optimization, complex-valued neural network, beamforming

1. Introduction

Neural networks have taken the centerstage in the field of machine learning and pattern recognition. The majority of reported work on networks, however, deals with *real-valued* networks – networks whose inputs, weights and activations are all entirely real. There are, however, several problems in which the ability to operate on complex input to produce real or complex-valued outputs is required. An example that easily comes to mind is that of signal processing problems that are ideally addressed in the complex short time Fourier transform (STFT) domain. If these operations were to be performed by a neural network, the network must deal with complex numbers. Instead, the common approach to these problems often is to ignore the phase (for example in denoising autoencoder applications by [1] and [2]) or to otherwise approximate the problem as one of real-valued calculus by simply splitting the backpropagation into two independent real-valued backpropagations ([3], [4]). Alternatively, the real and imaginary components of complex values may be stacked into a single real-valued array, *e.g.*, as in [5] who use this approach to estimate complex weights of an array beamformer from real-valued difference of arrival features or in [6], where a stacked output consists of a real and an imaginary part of a complex mask.

It is not that complex-valued neural networks (CVNNs) have not been proposed; indeed they have been around for a long time, *e.g.*,[7] or [8], who proposed CVNNs to be a generalization of real-valued neural networks (RVNNs) at least as early as 1992. Complex-valued variants of most typical neural network architectures have been proposed – as simple feed-forward networks, as auto-associative memories i.e. by [9], as recurrent networks ([10], [11]), etc. Not only have they been proposed for operating on complex-valued inputs, they have also been suggested as expansions of conventional real-valued networks that could potentially generalize their capabilities, such as by [8] above, and [12], etc. Several books, including recent ones such as [13] and [14] greatly summarize the specifics and issues related to CVNNs.

Nevertheless, it still remains unanswered, why complex-valued neural networks are not used for current speech enhancement algorithms, although these algorithms are typically formulated in the complex-valued STFT domain. Further, almost all works on CVNNs have been carried out in the pre-DNN era, leaving open the question if deeper CVNNs are a viable option.

This contribution tries to point out, which problems exist with complex-valued neural networks as well as advertises to still use real-valued neural networks for tasks which have traditionally been solved by algorithms in the complex domain. We selected a few speech enhancement problems and provide training results for these *minimal* examples both for real- and complex-valued neural networks. What sets this paper further apart from prior work is the introduction of the split rectified non-linearity, and the direct comparison of RVNNs and CVNNs.

2. Complex backpropagation

To motivate a gradient descent algorithm, we first need to clarify the differentiability problem. A complex function $f : \mathbb{C} \to \mathbb{C}$ is differentiable if the following limit converges to a single value independent of the path of h:

$$\frac{\mathrm{d}f}{\mathrm{d}z} = \lim_{h \to 0} \frac{f(z+h) - f(z)}{h} \tag{1}$$

An example that this does not hold for simple building blocks is given by the conjugate function $f(z) = z^*$ with two different paths for h, where $\eta \in \mathbb{R}$:

$$\lim_{\eta \to 0} \frac{(z+\eta)^* - z^*}{\eta} = 1, \quad \lim_{\mathrm{j}\eta \to 0} \frac{(z+\mathrm{j}\eta)^* - z^*}{\mathrm{j}\eta} = -1.$$

Only a certain class of functions is complex differentiable – these functions are called holomorphic. In contrast many relevant building blocks for neural networks, *e.g.* the cost function, can by definition not be holomorphic (due to its real-only output).

An elegant way around this is to make use of partial derivatives, where [15] nicely proved that non-holomorphic functions are still partially differentiable if one chooses the correct basis. In principle, we can use any non-degenerative rotation of the directions $x = \operatorname{Re} z$ or $y = \operatorname{Im} z$ as a basis. Using z and z^* is such a rotated basis and results in compact notations:

$$\mathrm{d}f = \frac{\partial f}{\partial z}\mathrm{d}z + \frac{\partial f}{\partial z^*}\mathrm{d}z^* \tag{2}$$

Although there is an obvious relation between z and z^*, we can still calculate partial derivatives by simply introducing auxiliary functions $a(z) = z$ and $b(z) = z^*$ and calculating partial derivatives with respect to them independently.

Using the above total derivative, we can now motivate a complex valued gradient descent as follows. Let l be the loss function of our network. The total differential is then given as follows:

$$\mathrm{d}l = \sum_k \frac{\partial l}{\partial z_k}\mathrm{d}z_k + \sum_k \frac{\partial l}{\partial z_k^*}\mathrm{d}z_k^* = \frac{\partial l}{\partial \mathbf{z}}\mathrm{d}\mathbf{z} + \frac{\partial l}{\partial \mathbf{z}^*}\mathrm{d}\mathbf{z}^*, \tag{3}$$

where the sum is over all output nodes.

If we now make use of $(\partial l/\partial \mathbf{z})^* = \partial l^*/\partial \mathbf{z}^*$ and $l = l^*$ (since l is real-valued) we can identify an inner product for the complex vector space:

$$\begin{aligned}\mathrm{d}l &= 2\operatorname{Re}\left\{\frac{\partial l}{\partial \mathbf{z}}\mathrm{d}\mathbf{z}\right\} = 2\operatorname{Re}\left\{\left(\left(\frac{\partial l}{\partial \mathbf{z}^*}\right)^{\mathsf{T}}\right)^{\mathsf{H}}\mathrm{d}\mathbf{z}\right\} \\ &= 2\operatorname{Re}\left\{(\nabla_{\mathbf{z}^*} l)^{\mathsf{H}}\mathrm{d}\mathbf{z}\right\}.\end{aligned} \tag{4}$$

We now follow the argumentation by [15]: Due to the Cauchy-Schwarz inequality $|\mathbf{a}^{\mathsf{H}}\mathbf{b}| \leq ||\mathbf{a}|| \cdot ||\mathbf{b}||$; the magnitude of the inner products of two vectors is bounded by the product of the norms. Therefore the inner product is maximized when the two vectors are colinear, i.e. $\mathbf{a} = \lambda\mathbf{b}$ where $\lambda \in \mathbb{C}$. Furthermore, the inner product is positive and real, when $\lambda \in \mathbb{R}_+$. Therefore, we conclude, that the function output l is maximally reduced, when we change $\mathbf{z}$ according to this update:

$$\mathrm{d}\mathbf{z} = -\mu\nabla_{\mathbf{z}^*} l, \quad \text{where } \mu \in \mathbb{R}_+. \tag{5}$$

To identify a recursion, we start by expanding the total derivative of $l(g(z))$ and reorganize the terms:

$$\begin{aligned}\mathrm{d}l &= \frac{\partial l}{\partial \mathbf{g}}\mathrm{d}\mathbf{g} + \frac{\partial l}{\partial \mathbf{g}^*}\mathrm{d}\mathbf{g}^* \\ &= \frac{\partial l}{\partial \mathbf{g}}\left(\frac{\partial \mathbf{g}}{\partial \mathbf{z}}\mathrm{d}\mathbf{z} + \frac{\partial \mathbf{g}}{\partial \mathbf{z}^*}\mathrm{d}\mathbf{z}^*\right) \\ &\quad + \frac{\partial l}{\partial \mathbf{g}^*}\left(\left(\frac{\partial \mathbf{g}}{\partial \mathbf{z}}\right)^*\mathrm{d}\mathbf{z}^* + \left(\frac{\partial \mathbf{g}}{\partial \mathbf{z}^*}\right)^*\mathrm{d}\mathbf{z}\right) \\ &= \left(\frac{\partial l}{\partial \mathbf{g}}\frac{\partial \mathbf{g}}{\partial \mathbf{z}} + \frac{\partial l}{\partial \mathbf{g}^*}\left(\frac{\partial \mathbf{g}}{\partial \mathbf{z}^*}\right)^*\right)\mathrm{d}\mathbf{z} \\ &\quad + \underbrace{\left(\frac{\partial l}{\partial \mathbf{g}}\frac{\partial \mathbf{g}}{\partial \mathbf{z}^*} + \frac{\partial l}{\partial \mathbf{g}^*}\left(\frac{\partial \mathbf{g}}{\partial \mathbf{z}}\right)^*\right)}_{= \frac{\partial l}{\partial \mathbf{z}^*} = (\nabla_{\mathbf{z}^*})^{\mathsf{T}}}\mathrm{d}\mathbf{z}^*\end{aligned} \tag{6}$$

Since l is a real-valued cost function $l = l^*$ and $(\partial l/\partial \mathbf{z})^* = \partial l^*/\partial \mathbf{z}^*$, the same equivalence holds: $\nabla_{\mathbf{z}^*} = (\nabla_{\mathbf{z}})^*$. Therefore, each network element has to calculate three different values:

$$\frac{\partial \mathbf{g}}{\partial \mathbf{z}}, \quad \frac{\partial \mathbf{g}}{\partial \mathbf{z}^*}, \quad \nabla_{\mathbf{z}^*} = \left((\nabla_{\mathbf{g}^*})^*\frac{\partial \mathbf{g}}{\partial \mathbf{z}^*} + \nabla_{\mathbf{g}^*}\left(\frac{\partial \mathbf{g}}{\partial \mathbf{z}}\right)^*\right). \tag{7}$$

Only the gradient $\nabla_{\mathbf{z}^*}$ then needs to be passed on to the previous network layer. This differs from [16], where they proposed to always propagate two different gradients because a possible objective might be complex. Since, for optimization problems like network training, a cost function always needs to be in a domain which is a totally ordered set (i.e. "$\leq$" is defined), the cost function will always be real-valued.

3. Building blocks

A feed forward neural network may now consist of fully connected layers, non-linearities and a loss function.

3.1. Linearity

A classic fully connected layer $\mathbf{g} = \mathbf{W}\mathbf{z} + \mathbf{b}$ with a complex parameter matrix $\mathbf{W}$ can be is used for training. It turns out, that an extended linear layer $\mathbf{h} = \mathbf{U}\mathbf{z} + \mathbf{V}\mathbf{z}^* + \mathbf{b}$, which holds a parameter matrix to transform both the input $\mathbf{z}$ and its conjugate $\mathbf{z}^*$ is not necessary, since the conjugate operation can be learned by just using classic fully connected layers as well. Therefore, we will simply use a classic fully connected layer with a single complex parameter matrix.

3.2. Non-Linearity

The Liouville theorem states that any bounded function that is also holomorphic must be a constant; if the function is bounded, but not a constant, it is not holomorphic. Different non-linearities, all of them being either unbounded, non-holomorphic, or both, have been presented in [14], *e.g.*:

$$f_{\mathrm{mt}}(z) = \tanh|z|\mathrm{e}^{\mathrm{j}\arg z}, \tag{8}$$

$$f_{\mathrm{st}}(z) = \tanh\operatorname{Re} z + \mathrm{j}\tanh\operatorname{Im} z, \tag{9}$$

where the former squashes the magnitude but does not change the phase. The latter operates on the real and imaginary part independently and is, thus, often called split non-linearity. Further non-linearities are summarized in [13, Section 1.5.1].

Motivated by the fact, that rectified linear units (ReLUs) work well for real-valued neural networks and are efficiently computed, we will rely on a split rectified linear units (SplitReLUs), meaning that the real and imaginary part is treated independently (see appendix for gradients). Preliminary experiments led to the conclusion, that this non-linearity is just as expressive as the aforementioned non-linearities, is faster to evaluate and easier to implement:

$$f_{\mathrm{sr}}(z) = \max(0, \operatorname{Re} z) + \mathrm{j}\max(0, \operatorname{Im} z). \tag{10}$$

3.3. Loss Function

Since, in the following, we want to compare real and complex valued neural networks (when applying both on complex valued problems) based on their loss function, we need to make sure, that the cost functions are identical.

The complex valued mean squared error (MSE) is calculated as follows:

$$l_{\mathrm{MSE}}^{\mathrm{complex}} = \frac{1}{K}(\mathbf{z}-\mathbf{t})^{\mathsf{H}}(\mathbf{z}-\mathbf{t}) = \frac{1}{K}\sum_{n=1}^{K}|z_k - t_k|^2, \tag{11}$$

where $\mathbf{z} \in \mathbb{C}^N$ and $(\cdot)^{\mathsf{H}}$ is the conjugate transpose.

Accordingly, the real-valued implementation is calculated as follows (corrected by a factor of 2):

$$l_{\text{MSE}}^{\text{real}} = \frac{1}{K}(\mathbf{z}^{'} - \mathbf{t}^{'})^{\mathsf{H}}(\mathbf{z}^{'} - \mathbf{t}^{'}) = \frac{1}{K}\sum_{n=1}^{2K}|z_k^{'} - t_k^{'}|^2, \quad (12)$$

where $\mathbf{z}^{'}, \mathbf{t}^{'} \in \mathbb{R}^{2K}$ are the stacked output of the real-valued neural network and the stacked training target, respectively.

In many tasks related to complex signal processing, the absolute phase as well as the scaling of the result may be less important than the level differences and the phase differences between signals. For example a beamforming vector $\mathbf{W}$ for a given frequency, which consists of complex scalars and is used to constructively compose an enhanced signal from STFTs of individual microphone signals serves just as well as any complex rotation $\mathbf{W}e^{\mathrm{j}\phi}$ of it.

Therefore, we use the negative cosine similarity (NCS) between complex vectors, where $||\mathbf{z}|| = \sqrt{\mathbf{z}^{\mathsf{H}}\mathbf{z}}$ is the vector length (see appendix for gradients):

$$l_{\text{NCS}}^{\text{complex}} = -\frac{|\mathbf{z}^{\mathsf{H}}\mathbf{t}|}{||\mathbf{z}|| \cdot ||\mathbf{t}||}, \quad l_{\text{NCS}}^{\text{real}} = -\frac{|\mathbf{z}^{'\mathsf{H}}\mathbf{t}^{'}|}{||\mathbf{z}^{'}|| \cdot ||\mathbf{t}^{'}||}. \quad (13)$$

4. Evaluation

To evaluate CVNNs in comparison to RVNNs, we decided to use a typical signal processing problem, namely beamforming, with a known analytic solution. We keep the degrees of freedom equal for both network types by setting the number of hidden units of the RVNN to be twice as high as for the CVNN. Stochastic gradient decent with a learning rate of 0.001 and a momentum of 0.9 was used in all training tasks.

We used 4620 and 1680 utterances from the TIMIT database [17] for training and cross validation, computed the STFT representation S_{tf} of clean speech, and multiplied it with a random complex-valued acoustic transfer function vector $\mathbf{H}_f$ in the sense of the multiplicative transfer function approximation. Each acoustic transfer function vector contains $D = 3$ complex random values and corresponds to the random (reverberant) transmission paths of the source signal to each of the D sensors. Additionally, the observations are corrupted with white noise $\mathbf{N}_{tf}$ with a signal-to-noise ratio (SNR) of $10\,\text{dB}$. The signal model is therefore:

$$\mathbf{Y}_{tf} = \mathbf{H}_f S_{tf} + \mathbf{N}_{tf}. \quad (14)$$

A classic approach is to use a principal component analysis (PCA) beamformer [18]. The PCA beamforming vector is obtained by first calculating the observation covariance matrix for each frequency bin:

$$\mathbf{\Phi}_{YY}(f) = \frac{1}{T}\sum_{t=1}^{T}\mathbf{Y}_{tf}\mathbf{Y}_{tf}^{\mathrm{H}}, \quad (15)$$

calculating the principal component $\mathbf{W}_f = \mathcal{P}\{\mathbf{\Phi}_{YY}(f)\}$ and finally applying the beamforming vector as a scalar product:

$$Z_{tf} = \mathbf{W}_f^{\mathrm{H}}\mathbf{Y}_{tf}. \quad (16)$$

Therefore, as a first task, we train both the real- and the complex-valued network to calculate the outer product $t = \mathbf{Y}_{tf}\mathbf{Y}_{tf}^{\mathrm{H}}$ of each observation vector. The loss function is set to calculate the MSE as given in Equation (12) for the real-valued and (11) for the complex valued neural network. The estimation for the outer product matrix (15) is treated independently for each time and frequency which effectively leads to a batch size of $F \cdot T$, where F is the number of frequency bins and T is the number of time frames. All further network parameters are summarized under Task 1 in Table 4.

Figure 1 shows the cross validation loss in terms of MSE (Equations (11), (12)) for the real-valued implementation in blue and the complex valued implementation in red for ten different initializations of the same network architecture (the factor 2 according to Equation (11) is taken into account). It can be observed, that the variance between different initializations increases during the observations. We can further note, that the loss for the real valued implementation is higher than for the complex valued network. How this affects SNR gains, is investigated subsequently.

In the second task, we train the two networks to calculate the principal component from the complex valued matrix $\mathbf{\Phi}_{YY}(f)$. Since the length of a principal component is arbitrary, we opted to use the negative cosine similarity as the loss function (Equation (13)). The other parameters are again listed in Table 4. It turns out that the difference in cross-validation score between the complex and real-valued networks is lower. It may also be observed that the loss is fairly close to the minimum of -1. Further improvement might be achieved by normalizing the input to reduce the dynamic range the network has to cover.

In the third task, we stack the previous networks to a deep architecture with the goal to estimate the beamforming vector $\mathbf{W}$. The training target is the oracle beamforming vector, which is obtained by calculating the principal component of the

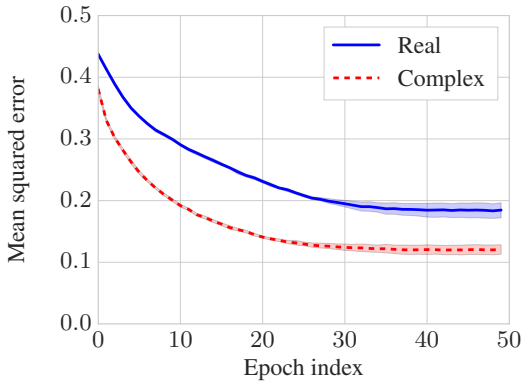

Figure 1: Summarized cross-validation error of 10 differently initialized networks to calculate the outer product matrix of TIMIT observations with the mean loss as a continuous line and the $95\,\%$ confidence intervals as a shaded area (Task 1).

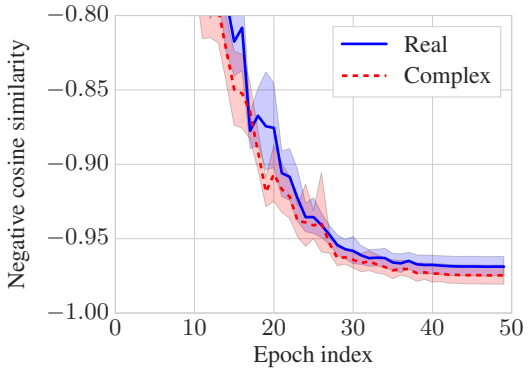

Figure 2: Summarized cross-validation error of 10 differently initialized networks to calculate the PCA vector given the covariance matrix of TIMIT observations visualized as in the first figure (Task 2).

	Task 1: Outer product		Task 2: PCA		Task 3: Beamformer	
	RVNN	CVNN	RVNN	CVNN	RVNN	CVNN
Layer 1	$2D \to 50$	$D \to 25$	$2D^2 \to 50$	$D^2 \to 25$	$2D \to 50$	$D \to 25$
Non-linearity	ReLU	SplitReLU	ReLU	SplitReLU	ReLU	SplitReLU
Layer 2	$50 \to 2D^2$	$25 \to D^2$	$50 \to 2D$	$25 \to D$	$50 \to 2D^2$	$25 \to D^2$
Layer 3					$2D^2 \to 50$	$D^2 \to 25$
Non-linearity					ReLU	SplitReLU
Layer 4					$50 \to 2D$	$25 \to D$
Loss function	MSE	MSE	NCS	NCS	NCS	NCS

Table 1: Network configurations for the different tasks and the real and the complex neural network.

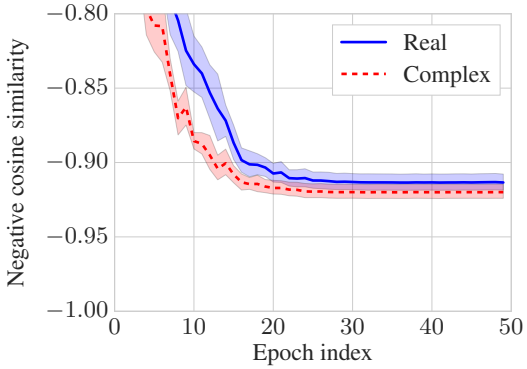

Figure 3: Summarized cross-validation error of 10 differently initialized networks to calculate the PCA vector given noisy TIMIT observations visualized as in the first figure (Task 3).

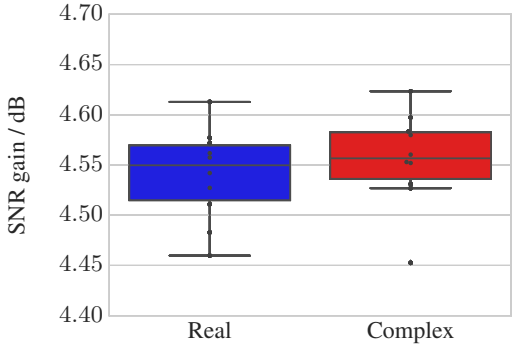

Figure 4: Boxplot of the median SNR gain for all simulations performed with a real-valued and a complex-valued neural network (Task 3).

speech-only covariance matrix. Thus, the neural network may find a better beamforming vector than simply calculating the principal component of the noisy observation covariance matrix. For the first two layers of the network, each feature vector is an observation $\mathbf{Y}_{tf} \in \mathbb{C}^D$. Therefore, this part of the network operates independently on each time-frequency slot. The output of the second layer is then summarized along time:

$$\mathbf{h}_f^{'} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{h}_{tf}. \tag{17}$$

To some extend, the first two layers and the summation are expected to provide information similar to the covariance matrix output of Task 1. The following two layers then calculate the principal component similar to Task 2. It turns out that the differences between the real and the complex networks are just as small as during Task 2. To understand how these loss values translate into SNR gains, Figure 4 evaluates the enhancement performance when using the estimated beamforming vector and Equation (16).

In all these experiments the number of real-valued parameters was the same for the RVNN and CVNN, meaning that the number of complex parameters is half the number of real parameters. Nevertheless, due to the complex multiplications during the forward and backward steps, a CVNN needs more real-valued multiplications than a RVNN (more elementary operations in general). The amount of real-valued additions and the amount of real-valued comparisons (due to the rectified units) are the same for both networks.

5. Conclusions

From the presented experiments, we conclude that complex valued neural network do not perform dramatically better than real-valued networks on the considered tasks, which are intrinsically complex. Still, they require more multiplications. We therefore conclude that we can rely on real-valued implementations and use the real and imaginary part of features as inputs, if complex regression problems are to be solved. Nevertheless, we promote the negative cosine loss as a means of penalizing errors in relative phase and level differences, which provides the desired invariance with respect to an absolute phase and scaling.

6. Appendix

The gradients for the SplitReLU are given by

$$\frac{\partial f_{\mathrm{sr}}}{\partial \operatorname{Re} z} = [\operatorname{Re} z > 0], \quad \frac{\partial f_{\mathrm{sr}}}{\partial \operatorname{Im} z} = \mathrm{j}[\operatorname{Im} z > 0], \tag{18}$$

$$\nabla_{\mathbf{z}^*} = \operatorname{Re}\left\{\nabla^*_{f_{\mathrm{sr}}^*} \frac{\partial f_{\mathrm{sr}}}{\partial \operatorname{Re} z}\right\} + \mathrm{j}\operatorname{Re}\left\{\nabla^*_{f_{\mathrm{sr}}^*} \frac{\partial f_{\mathrm{sr}}}{\partial \operatorname{Im} z}\right\}. \tag{19}$$

The gradient of the NCS can be composed of atomic computations, where most notably, the gradient of the absolute value function $a(z) = |z|$ is given as follows:

$$\nabla_{z^*} = \frac{1}{2}\mathrm{e}^{\mathrm{j}\arg z}\left(\nabla^*_{a^*} + \nabla_{a^*}\right) \tag{20}$$

7. Acknowledgements

Supported by Deutsche Forschungsgemeinschaft under contract no. Ha3455/12-1 and the DFG-NSF collaboration within the Priority Program SPP1527 “Autonomous Learning”.

8. References

[1] S. Araki, T. Hayashi, M. Delcroix, M. Fujimoto, K. Takeda, and T. Nakatani, "Exploring multi-channel features for denoising-autoencoder-based speech enhancement," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 116–120.

[2] H. Erdogan, J. R. Hershey, S. Watanabe, and J. Le Roux, "Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[3] H. Zhang, C. Zhang, and W. Wu, "Convergence of batch split-complex backpropagation algorithm for complex-valued neural networks," *Discrete Dynamics in Nature and Society*, 2009.

[4] D. Xu, H. Zhang, and L. Liu, "Convergence analysis of three classes of split-complex gradient algorithms for complex-valued recurrent neural networks," *Neural computation*, vol. 22, no. 10, pp. 2655–2677, 2010.

[5] X. Xiao, S. Watanabe, H. Erdogan, L. Lu, J. Hershey, M. L. Seltzer, G. Chen, Y. Zhang, M. Mandel, and D. Yu, "Deep beamforming networks for multi-channel speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[6] D. S. Williamson, Y. Wang, and D. Wang, "Complex ratio masking for joint enhancement of magnitude and phase," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[7] H. Leung and S. Haykin, "The complex backpropagation algorithm," *IEEE Transactions on signal processing*, vol. 39, no. 9, pp. 2101–2104, 1991.

[8] A. Hirose, "Proposal of fully complex-valued neural networks," in *IEEE International Joint Conference on Neural Networks (IJCNN)*, vol. 4, 1992, pp. 152–157.

[9] S. Jankowski, A. Lozowski, and J. M. Zurada, "Complex-valued multistate neural associative memory," *IEEE Transactions on Neural Networks*, vol. 7, no. 6, pp. 1491–1496, 1996.

[10] D. Mandic and J. Chambers, *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*, ser. Adaptive and Learning Systems for Signal Processing, Communications and Control Series. Wiley, 2001.

[11] M. Yoshida and T. Mori, "Global stability analysis for complex-valued recurrent neural networks and its application to convex optimization problems," *Complex-Valued Neural Networks: Utilizing High-Dimensional Parameters*, pp. 104–114, 2009.

[12] M. F. Amin and K. Murase, "Single-layered complex-valued neural network for real-valued classification problems," *Neurocomputing*, vol. 72, no. 4, pp. 945–955, 2009.

[13] T. Adali and S. Haykin, *Adaptive signal processing: next generation solutions*. John Wiley & Sons, 2010, vol. 55.

[14] A. Hirose, *Complex-valued neural networks (2nd Edition)*. Springer Science & Business Media, 2012.

[15] D. H. Brandwood, "A complex gradient operator and its application in adaptive array theory," in *IEE Proceedings F (Communications, Radar and Signal Processing)*, vol. 130. IET, 1983, pp. 11–16.

[16] M. F. Amin, M. I. Amin, A. Al-Nuaimi, and K. Murase, "Wirtinger calculus based gradient descent and Levenberg-Marquardt learning algorithms in complex-valued neural networks," in *Neural Information Processing*. Springer, 2011, pp. 550–559.

[17] J. S. Garofolo, L. F. Lamel, W. M. Fisher, D. S. Pallett, and N. L. Dahlgren, *The DARPA TIMIT Acoustic-Phonetic Continous Speech Corpus CDROM*, U.S. Department of Commerce, 1993.

[18] E. Warsitz and R. Haeb-Umbach, "Acoustic filter-and-sum beamforming by adaptive principal component analysis," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2005.