Hybrid Real- and Complex-valued Neural Network Architecture

Paper #2366

Abstract. We propose a *hybrid* real- and complex-valued *neural network* (HNN) architecture, to combine the advantages of real- and complex-valued data processing. Using an example real-valued neural network (RVNN) for solving an inherently complex-valued problem, we show how it learnt to do complex-valued convolution. Yet, we highlight its inefficiency stemming from the constraint of the fully real-valued processing. To create the HNN, we propose to use building blocks containing both real- and complex-valued paths, where information between domains is exchanged through domain conversion functions. We also propose the use of novel complex-valued activation functions, with higher generalisation and parameterisation efficiency. Furthermore, a HNN-specific architecture search framework has been developed to reduce the dimensionality of the network. We verify the utility of the HNN in comparison to a real-valued architecture with an experiment using the AudioMNIST dataset. In particular, the HNN reduced the cross entropy loss by a factor of 3.8 compared to the RVNN even with slightly lower parameter storage requirements. Such results show great promise for the use of partially complex-valued processing in neural networks.

1 Introduction

The inherent necessity for complex-valued data processing, in many fields of science where deep learning is applied, requires data to be converted into a real-valued representation. For example, the phase component of a complex-valued audio spectrum is often discarded. However, by doing this, the audio becomes less intelligible and the late reconstruction of the phase limits the quality of frequency-domain audio synthesis [14]. When the phase is not discarded, but the real and imaginary parts of a complex-valued signal are split and processed independently, additional phase distortion can occur [9].

Deep complex-valued neural networks (NNs) have been applied recently to solving traditionally complex-valued problems, e.g., radar [5] and audio processing [20], and even real-valued problems like image processing [13], presenting performance gains over RVNNs in most cases. Moreover, complex-valued processing in NNs has shown richer representation capacity with better generalisation characteristics, additionally allowing for faster learning and noise-robust memory mechanisms [17]. Automatic differentiation and other overheads within complex-valued neural networks typically incur an increased training time compared to using standard functions within a RVNN [9].

The combination of real- and complex-valued processing, in neural network architectures, was previously considered but scarcely explored. Averaging the results of a RVNN and its complex-valued counterpart has been investigated [12] – possibly the simplest approach towards the idea of a hybrid real- and complex-valued architecture. On the other hand, Du et al. [4] created a network with complex-valued layers followed by real-valued layers. In both cases, the hybrid version displayed better results than real- or complex-valued counterparts. A two-stage framework has also been explored [8], where a RVNN is followed by a complex-valued processing stage. Similarly, Lv et al. [10] applied additional real-valued processing to a complex-valued neural network. However, the interchange of information between both domains was not explored and no systematic approach was provided in the design of a hybrid architecture. For this reason, one of the main contributions of our work is to propose a comprehensive framework for designing HNNs.

We present a novel and flexible approach that attempts to combine, in a systematic way, the "best of both worlds". In particular, the proposed neural network architecture has both real- and complex-valued pathways, allowing us to choose the appropriate functions and processing in each domain to minimise loss and parameter storage. We also devised a Neural Architecture Search (NAS) procedure such that redundancies between paths can be reduced together with a decrease in the number of parameters[1] and operations.

Specifically, in Section 2 we show how we arrived at the proposed HNN architecture by noticing the emergence of (partial) complex-valued processing within a RVNN. This prompted us to design HNN blocks that are very general, elevating the number of parameters significantly in the initial stage of the design but giving the network a platform to solve the problem. In the functional blocks, information is interchanged between both domains with domain conversion functions. To reduce the number of parameters, in Section 3 we also propose a NAS procedure for model optimisation. Furthermore, we outline classic domain conversion functions and propose new ones based on further observations of the RVNN in Section 4. Novel complex-valued activation functions are elaborated in Section 5, aiming for a reduction of required processing power and higher parameterisation efficiency.

Section 6 presents numerical experiment, where we verified the viability of the HNN using the AudioMNIST dataset as a classification problem. It consists of inputting the short-term Fourier transform (STFT) of the audio signal to the NN, obtaining the label of the identified digit at the output. We optimised performance using the proposed NAS procedure for both RVNN and HNN. By comparing HNN and RVNN results, we observed a significant reduction in the cross entropy loss over the test set – a factor of 3.8. This result holds great promise for the partial use of complex-valued processing in neural networks. Finally, we present a discussion over the results and methods in Section 7 and conclude the paper in Section 8.

[1] Note that, when referring to parameters, we count two parameters per complex weight or bias and one per real weight or bias.

2 Architecture

The HNN architecture was inspired by observations of complex-valued processes in RVNNs when dealing with complex-valued problems. A complex-valued input was connected to a RVNN by splitting and interleaving the real and imaginary parts. By plotting the weights of a fully connected layer in this network and re-ordering the outputs by similarity, we noticed a weight value behaviour that could be translated into a complex-valued convolution. Next, we describe a straightforward example problem where this behaviour can be observed.

2.1 Example experiment

Consider a signal model

$$y = ae^{i(xm+p)}, \quad x = \frac{2\pi n}{N} - \pi, \quad n = 0...(N-1), \qquad (1)$$

where a is the amplitude, m is the frequency in cycles and p is the phase in radians. Given the discrete Fourier transform (DFT) of $y+v$, with v being a uniformly distributed (in magnitude and phase) zero-mean complex-valued noise signal, retrieve the signal model parameters by estimating $[m, a\sin(p), a\cos(p)]$.

The signal model is uniformly sampled at n, providing N values of y, these are element-wise multiplied by a Hanning window function and input to a DFT of N points. For this example, $N = 512$ points. The complex-valued frequency bins 0 (DC) to 17 of the DFT output are used, of which the real and imaginary parts are split and interleaved, forming 36 real-valued predictors. For sample generation, the parameters are randomly drawn from uniform distributions, with ranges: $a \in \{0.1, 1.0\}$; $m \in \{5.0, 12.0\}$; $p \in \{-\pi, \pi\}$. The maximum noise amplitude is 0.01. A training set is generated with 500k samples.

Moreover, we defined a neural network model consisting of four fully connected layers, where the first three are followed by an exponential linear unit (ELU) activation function with hyperparameter $\alpha = 1$. The output sizes of the four layers of the network are 18, 14, 4 and 3, respectively. The NN model was trained multiple times to verify that it exhibits similar weight behaviour. Figure 1 shows the weights of the first fully connected layer mapped from input to output, obtained after 2500 epochs of training and normalised from -1 to 1, where the first column is the bias. No relevant conclusions can be directly drawn from such a plot, except from the insight that some outputs follow a similar pattern. By reordering the outputs, placing similar weight rows together, as per Figure 2, we can observe a very strong pattern in the weight values, mainly from outputs 1 to 10.

To better understand the pattern from the example, we zoom into a small section of Figure 2, obtaining Figure 3, which is analysed next. Let $\boldsymbol{in} = [in_{11} \;\; in_{12}]^T$, where the superscript ^{T} indicates the transpose operation, represent the input signal vector for inputs 11 (real) and 12 (imaginary). The weights with values depicted in Figure 3 map inputs 11 and 12 to an output vector $\boldsymbol{out} = [out_3 \;\; out_4]^T$ with a weight matrix

$$\boldsymbol{W} = \begin{bmatrix} 0.827 & -0.986 \\ 0.986 & 0.819 \end{bmatrix}. \qquad (2)$$

We can define the output as

$$\begin{bmatrix} out_3 \\ out_4 \end{bmatrix} = \boldsymbol{W} \times \boldsymbol{in} = \begin{bmatrix} 0.827 & -0.986 \\ 0.986 & 0.819 \end{bmatrix} \times \begin{bmatrix} in_{11} \\ in_{12} \end{bmatrix}. \qquad (3)$$

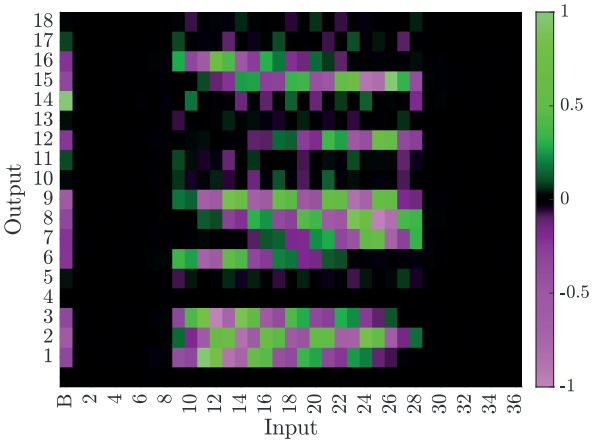


Figure 1: Weights of the first fully connected layer.

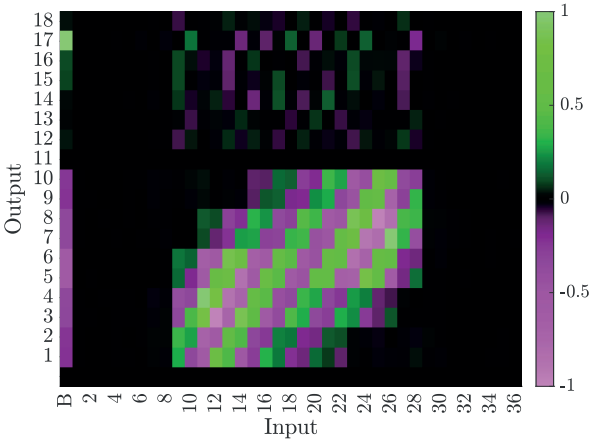


Figure 2: Reordered weights of the first fully connected layer.

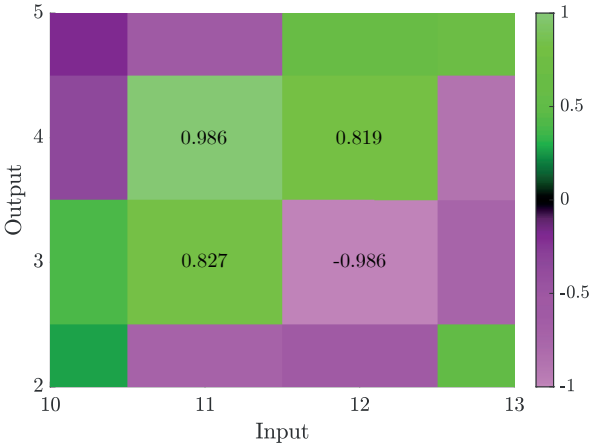


Figure 3: Region of reordered weights showing conjugate-value relations.

On the other hand, let us define $x = x_r + ix_i$ and $w = w_r + iw_i$ as complex numbers. The complex-valued convolution between x and w results in a complex number $o = o_r + io_i$ which can be obtained as $o = w * x$, where

$$\begin{bmatrix} o_r \\ o_i \end{bmatrix} = \begin{bmatrix} w_r & -w_i \\ w_i & w_r \end{bmatrix} \times \begin{bmatrix} x_r \\ x_i \end{bmatrix}. \qquad (4)$$

With the similarity of (3) and (4), out_3 and out_4 are equivalent to o_r and o_i, as well as in_{11} and in_{12} to x_r and x_i. Therefore, we

can notice that $\boldsymbol{W}$ contains a good approximation of the complex conjugates of a complex number. In this sense, we can interpret (3) as an approximate representative part of a complex-valued convolution.

This tells us that the neural network, in part, is inherently learning to approximate a complex-valued convolution using only real-valued weights. It is inefficient for the real-valued network to recreate complex-valued convolutions as four real-valued weights are required for each complex-valued multiplication, whereas in an efficient complex-valued multiplication, only two weights are required with the associated reduction in memory space and access. Additionally, outputs 11 to 18 in Figure 2 do not show such patterns consistent with complex-valued convolution, simply representing a real-valued convolution. In this way, we can observe a necessity of adding the ability of complex processing to the neural network, while still allowing for real-valued paths.

Using the trained network for inference with generated datasets that kept the amplitude and frequency model parameters constant but swept the phase parameter between $-\pi$ and $+\pi$, we were able to observe how a rotating phasor at the output was internally conveyed. We saw that 4- and 3-phase activations with a positive value were present as the information was transformed into the phasor outputs. This presented the first insights into how a complex value could be mapped into a multi-dimensional positive real space and vice-versa. Such a mapping is taken into account as a possible conversion function and outlined in Section 4.

Allowing the network to have paths of complex-valued processing while still maintaining real-valued branches would be a natural solution to facilitate complex-valued processing and avoid the disadvantages that a purely complex-valued model would bring. Next, we present a hybrid real- and complex-valued building block for convolutional/fully connected neural networks, which we use to construct the HNN model.

2.2 *Building block*

The HNN is built upon a functional block with real- and complex-valued inputs and outputs. For generalisation, separate pathways allow all inputs to connect to all outputs using real-to-complex and complex-to-real domain conversion functions. Such functions facilitate the information flow between the two domains and are explored in detail in Section 4. Each path, real- or complex-valued, can have a convolution with an activation function and additional (optional) functions such as pooling, normalisation and dropout. Complex-valued paths use complex-valued functions and real-valued paths use real-valued functions.

The diagram of a functional block can be seen in Figure 4, where inputs and outputs are numbered for further simplification. The "concat" rectangles represent concatenations, while "conv" stands for convolution, σ contains the activation function and other optional functions and the hourglass-shaped blocks ($\mathbb{C} \rightarrow \mathbb{R}$ and $\mathbb{R} \rightarrow \mathbb{C}$) are the domain conversion functions. In the figure, yellow-coloured blocks stand for real-valued processing and blue-coloured blocks stand for complex-valued processing. Importantly, the use of multiple pathways allows for generalisation of the architecture, which can be optimised according to the procedure described in Section 3. The HNN is formed by multiple building blocks, connecting corresponding real- and complex-valued inputs and outputs together in series and/or parallel. In this work, we specifically try to build a network by connecting such blocks in series.

If the system input data is only real- or only complex-valued, then the unused ports can be either removed – with the associated routing

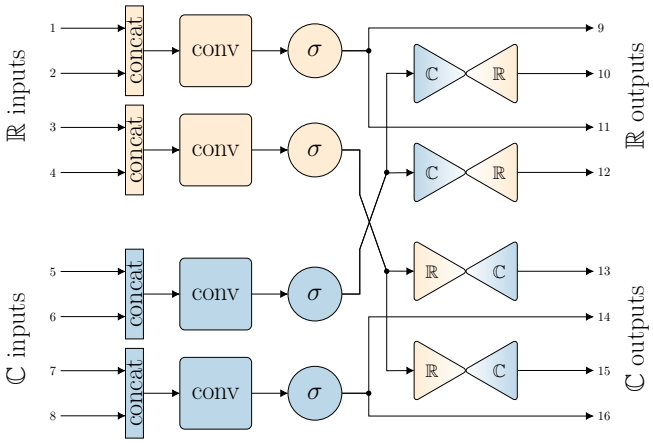


Figure 4: HNN functional block.

in the next block – or domain conversions can be performed from the used input to the other domain input. Equivalently, if the system outputs are not required to be both real- and complex-valued, the unused output can be removed with all dependent routing.

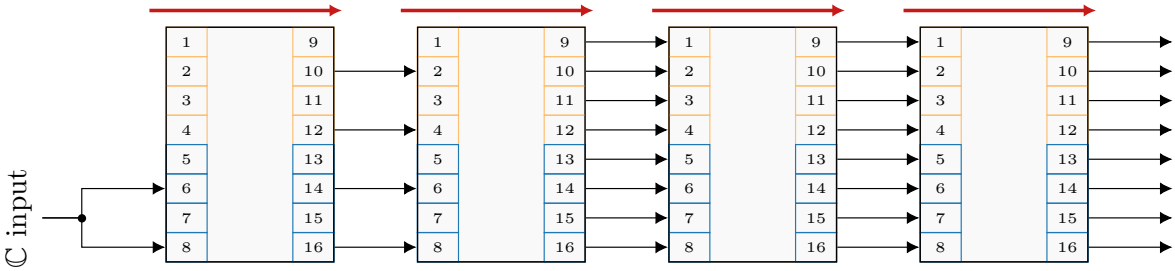
Furthermore, the initial step in designing a HNN is to form a base architecture composed of multiple building blocks. Such an architecture can be inherited from a real-valued NN model, substituting each convolutional/fully connected layer for the functional block in Figure 4, or defined from scratch with multiple interconnected blocks.

The architecture can be tailored for the problem at hand, in the case that the type of input and output are available as prior knowledge. For example, if the input is the STFT of a signal, the complex-valued inputs 5 and 7 or 6 and 8 from Figure 4 could be used, including domain conversion functions that would transform the complex-valued input into a real-valued input, applied to inputs 1 and 3 or 2 and 4 from the functional block. Propagating forward through such connections in the HNN model determines which functions are used. Equivalently, the same can be done for the outputs. Considering, e.g., a real-valued output, a backward dependency check can be done in the hybrid model to remove unnecessary connections. An example of the forward and backward pass for only complex-valued input and real-valued output, for a four-layer HNN, can be seen in Figure 5, where the building block from Figure 4 is represented by a unit containing eight inputs and eight outputs that match the functional block.

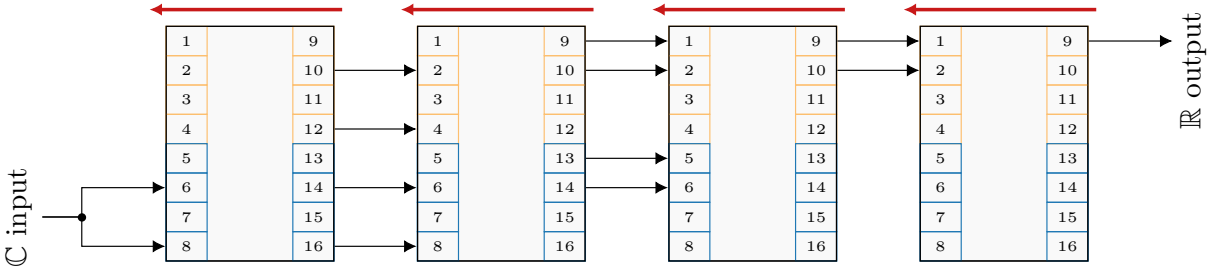
By defining a base architecture and checking dependencies, we arrive at an initial model which will be subject to a systematic architecture search procedure. This step is needed for reducing the total number of parameters and required processing that the initial hybrid model presents and thus making it feasible for practical implementation. Importantly, the search also automates the choice of activation and domain conversion functions, which is a time-consuming task if performed manually. The next section presents our proposed architecture search scheme for HNNs.

3 Architecture search

The aforementioned neural network block is used for building HNNs. To find an optimal trade-off between performance and complexity, fine-tuned parameters and an efficient network structure are required. Conventionally, this process has relied heavily on time consuming manual design, however, with developments in NAS methods, automated search algorithms are increasingly used. In particular, Optuna, originally developed by Akiba et al. [1] in 2019, is a cutting-edge hyperparameter optimisation framework which returns an optimised



(a) Forward pass



(b) Backward pass

Figure 5: Architecture dependencies check example.

neural network configuration by evaluating the performance of different structures and hyperparameters. Thus, we adopted a framework for building HNNs using Optuna. The methodology comprises eight sequential phases aimed at tailoring and refining a HNN to suit a specific task. The sequence of phases is shown in Figure 6 and is described next.

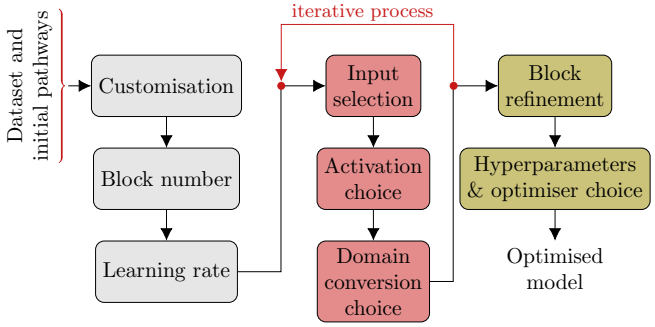


Figure 6: HNN architecture search process.

Customisation: the input and output pathways are adjusted to fit the specific needs of the task. Subsequently, as mentioned earlier in Subsection 2.2 and exemplified in Figure 5, dependencies between various blocks need to be checked to establish the default architecture and to initially mitigate redundancy. The default network is used as a baseline in the NAS process.

Block number: the number of blocks in the neural network architecture is determined based on performance. As discussed, the functional blocks are the modular components that form the network structure. Deciding on the appropriate number of blocks is essential for balancing model complexity and computational efficiency.

Until the block number number selection, the optimisation objective regards only validation loss reduction. For the subsequent steps, a limitation in the number of parameters is also included as part of the objective.

Learning rate: during this phase, the preliminary learning rate is established. This requires the selection of an initial learning rate that serves as a starting hyperparameter for further adjustment. The learning rate has a strong influence on the optimisation convergence.

The next few phases are related to the selection of paths, activation functions, and domain conversion functions for block N_b, where N_b is the index of the block. This process contains three sequential phases, iteratively executed for each block within the architecture.

Input selection: for the purpose of reducing redundancy, identify unnecessary paths in all the blocks. Whenever any architecture path change takes place, dependencies are evaluated again to determine the connectivity of paths across distinct blocks. These paths determine the flow of information into the block, thereby influencing the feature extraction process during training.

Activation choice: activation functions play an important role by introducing non-linearity into the neural network. In this phase, the activation functions for block N_b are chosen from a candidate list. We also offer the option of "no activation" as a selection for functions that precede a domain conversion function, since they can also introduce non-linear transformations.

Domain conversion choice: we select domain conversion functions from a candidate list, which serve to interchange information between real- and complex-valued pathways.

Block refinement: this step involves the optimisation of blocks within the architecture, including adjustments in convolution channel number and whether other function layers should be included within each block. The inclusion of specific layers and their corresponding hyperparameters are explored. This optimisation phase aims to enhance the network's ability to discover relevant features from the input data while mitigating computational costs.

Hyperparameters & optimiser choice: the last step includes the optimisation of the learning rate and the selection of a better optimisation algorithm, aimed at enhancing convergence speed and overall efficiency.

This methodology facilitates the comprehensive building and optimisation of HNN architectures tailored to a specific task.

4 Domain conversion

HNNs uniquely require conversion between the real- and complex-domains. This is usually limited to Cartesian or polar conversions. In the example experiment described in Section 2, we also observed that other functions may be useful and thus we widened the function space. For the purposes of building a HNN, the mapping does not need to be invertible and, for generalisation, should be allowed to contract or expand dimensionality with the associated loss or redundancy in information.

In that sense, we define three different forms of conversion from real- to complex-valued domain, mostly based on Cartesian and polar functions, shown in the following. An expansive form, where a single real-valued x_1 input is converted into a complex-valued output z as

$$z = x_1; \quad z = e^{ix_1}; \quad z = |x_1|e^{ix_1}, \tag{5}$$

which we name "Real", "Exp" and "MagExp", respectively, for further use; an equivalent transformation, where two different real-valued inputs x_1 and x_2 are converted into a complex number, representing either real and imaginary parts or amplitude and angle, as

$$z = x_1 + ix_2; \quad z = x_1 e^{ix_2}, \tag{6}$$

which are the "Cartesian" and "Polar" transformations; and a reduction form, where three real-valued inputs x_1, x_2 and x_3 are converted into a single complex-valued output by the "Rotation" function

$$z = (x_1 + ix_2)e^{ix_3}. \tag{7}$$

The functions with two real-valued inputs are a good choice for the primary network architecture search to reduce the search space as they neither lose information nor introduce redundancy.

We also define a set of complex to real domain conversion functions for improved handling of phase information as well as producing multiple outputs. Raw phase information is generally not well tolerated in gradient descent-based systems due to the discontinuity as the phase value wraps around $\pm\pi$. For this reason, we propose to use the absolute value of the phase, eliminating the discontinuity, resulting in the following reduction (information loss) conversion functions:

$$y = \Re(z); \quad y = |z|; \quad y = |z|^2; \quad y = |\angle(z)|/\pi, \tag{8}$$

named respectively "Real", "Mag", "SquareMag" and "AbsPhase"; and

$$\{y_1 = |z|,\ y_2 = |\angle(z)|/\pi\}, \tag{9}$$

which we call "MagAbsPhase", where $\angle$ represents angle, y is the real-valued output, y_1 and y_2 are two real-valued outputs when there is more than one output in a conversion function and z is a complex number used as input. Additionally, we also consider the following lossless conversion:

$$\{y_1 = \Re(z),\ y_2 = \Im(z)\}, \tag{10}$$

which is commonly used for real-valued representations of complex numbers.

The second case was inspired by observations in Section 2, to reproduce the positive biased functions, a combination of the magnitude (providing positive offset) and a phase-shifted version of the complex-valued input was used. By equally spacing the phase shifts, any number of outputs can be produced by the "MultiMagReal" function:

$$y_n = \frac{(|z| + \Re(ze^{-2i\pi n/N_p})}{2}, \quad n = 0...(N_p - 1), \tag{11}$$

in which all output values $y_n \geq 0$, N_p is the number of output phases and $N_p \geq 3$ allows for a lossless conversion. In a similar way, we can also define an angle-based conversion function "MultiMagPhase" as

$$y_n = \frac{|z||\angle(ze^{-2i\pi n/N})|}{\pi}, \quad n = 0...(N_p - 1). \tag{12}$$

For both functions (11) and (12), $N_p = 3$ is a fair choice, reducing the preliminary network architecture search space and maintaining a conversion without loss of information.

5 Complex-valued activation functions

Many complex-valued activation functions have been proposed [17, 2]. Starting from the straightforward function $Z_o = z/(|z| + \epsilon)$ [6], where ϵ is a small positive number, we first generalised it to include a soft threshold shape when z is real-valued, by including α and β parameters, resulting in $Z_o = \alpha z + \beta z/(|z| + \epsilon)$. This also gave us the first hint that similar functions in this class can be used effectively for both real- and complex-valued arguments. The following proposed functions are further generalisations:

$$Z_o = \alpha z + z^p \frac{\beta + \gamma z}{|z|^q + \epsilon}; \tag{13a}$$

$$Z_o = \alpha z + z^p \frac{\beta + \gamma z}{\sqrt{|z|^q + \epsilon}}; \tag{13b}$$

$$Z_o = \alpha z + \frac{\beta + \gamma z^p}{\sqrt{|z|^q + \epsilon}}. \tag{13c}$$

where p and q are integers from 0 to 4, α, β, γ and ϵ are real numbers and $\epsilon > 0$.

The activation function in (13a) includes approximations or substitutes for the rectified linear unit (ReLU), exponential linear unit (ELU) variants, Tanh shrink and absolute real-valued activation functions. For example, the ELU activation can be approximated with $Z_o = 0.373z + z(1.513 + 0.627z)/(|z| + 2.411)$. For better visualisation, the real-valued ELU function and its approximation are presented in Figure 7 and show close agreement over the range of $x \in \{-3, 3\}$, while the complex-valued ELU function is presented in Figure 8 with an input stimulus (spiral) of $z = xe^{2i\pi x}$, $x \in \{0, 3\}$. Functions (13b) and (13c) include the real-valued Tanh- and Softplus-shaped functions, respectively. Many familiar real-valued functions can thus be extended to the complex domain via these proposed general forms.

Additionally, by defining

$$v = \Re(z) - \beta|\Im(z)| - \gamma, \tag{14}$$

we can form two purely complex-valued (no real-valued equivalents) activation functions to add diversity to the function library:

$$Z_o = z(1 - |\alpha| + \alpha \max(0, \text{sign}(v))), \tag{15}$$

which is a complex-valued switching function as another analogue to the real-valued ReLU function; and

$$Z_o = z\left(1 - |\alpha| + \frac{\alpha \max(0, v)^p}{|z|^q + \epsilon}\right), \tag{16}$$

which has a smooth transition instead.

To reduce the search space during network optimisation, seven basic shapes of complex-valued functions are considered, where five relate to real-valued equivalent functions: ReLU, Tanh, TanhShrink,

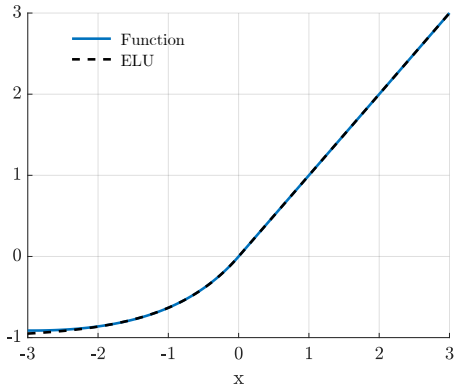


Figure 7: ELU function approximation.

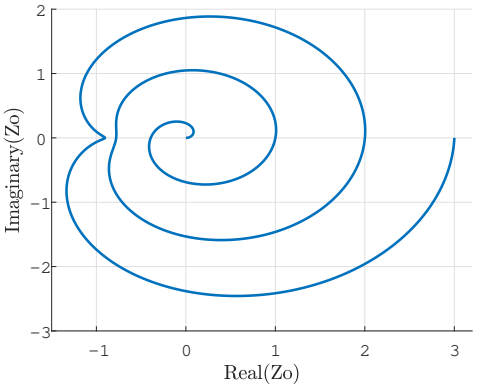


Figure 8: Complex-valued ELU function.

SoftPlus and Abs (an uncommon activation function, identified in [15]); and two cases for (15) and (16). The parameters p, q, α, β, γ and ϵ used to achieve the aforementioned shapes are shown in Table 1, with their proposed names. The parameters of the chosen activations can then be tuned in later optimisation processes if desired.

Table 1: Parameters used to define the shape of the considered complex-valued activation functions.

Name	Eq.	p	q	α	β	γ	ϵ
cReLU	(13a)	1	1	0.500	0.000	0.500	0.010
cAbs	(13a)	1	1	0.000	0.000	1.000	0.010
cTanhShrink	(13a)	3	2	0.000	1.000	0.000	1.000
cTanh	(13b)	1	2	0.000	1.000	0.000	1.000
cSoftPlus	(13c)	2	2	0.500	2.134	0.500	3.079
cReImLU	(15)	-	-	0.950	1.000	0.100	-
cRecipMax	(16)	2	2	-0.900	0.500	0.100	0.100

5.1 Dual input activations

In addition, we have defined two-input (z and u) complex-valued activation functions to perform multiplication: $Z_o = zu$ and $Z_o = z\overline{u}$ ($\overline{u}$ represents the complex conjugate of u); and a form of division for z/u that can handle a zero value denominator:

$$Z_o = z\frac{\overline{u} + \epsilon}{|u|^2 + \epsilon^2}. \tag{17}$$

When $|u| \gg \epsilon$, $Z_o \to z/u$ and when $|u| = 0$, $Z_o = z/\epsilon$. Naturally, ϵ should not be too small so that Z_o is limited to be within a predictable range to prevent the network from becoming unstable, e.g., $\epsilon = 0.1$, then $|Z_o| \leq 10|z|$. To form two sets of inputs, the incoming data must be equally split. These functions can be candidates in later stage network optimisations.

6 Numerical experiment

In order to verify the practicality and advantages of the HNN model, we utilised the AudioMNIST [3] dataset as a proof of concept. The dataset includes 30k audio samples, consisting of the spoken digits (0-9) uttered by 60 distinct speakers. Employing Optuna optimisation techniques, the conventional real-valued neural network was refined to be compared against the proposed HNN, which was constructed according to previously outlined methodologies. The primary objective is to evaluate the performance of both models in the classification of spoken digits.

The raw audio data from AudioMNIST at 48k samples/s rate is normalised and zero-padded to 1 second of duration, where the audio content is time-shifted with a random offset for each access to satisfy the time invariance requirement. An STFT of 960 points and 50% overlap using a Hanning window is applied to the signal providing a time and frequency grid resolution of 10 ms and 50 Hz, respectively. The real and imaginary parts of the STFT are interleaved in the frequency dimension, resulting in a real-valued input. Magnitude compression of 0.5 was applied to the input data ($Z_o = |z|^{0.5}e^{i\angle(z)}$) to reduce dynamic range without altering the phase. 10% of the dataset is randomly separated for a validation set, and another 10% for a testing set. For each audio file at the input, the output label is a class from 0 to 9.

As a baseline, we consider the RVNN model from [18], simplified to have convolutions operating in the frequency dimension of the input data – convolution kernel in the form of $(m, 1)$, where m is the kernel size in the frequency dimension. For a fair comparison, we applied the same architecture search procedure as described in Section 3. We let Optuna select the number of modules, convolutional and fully connected output channels, layer combinations inside each module (whether to use dropout, max pooling and group normalisation), the activation functions and other important hyperparameters. The considered activation functions in the real-valued architecture search procedure were: ReLU, PReLU, SELU, CELU, abs, soft shrink, hard shrink, tanh shrink, sin, sinh, cosh, hard tanh and exp. The optimised architecture is shown in Figure 9, achieving a cross entropy loss of 0.020, also included in Table 2.

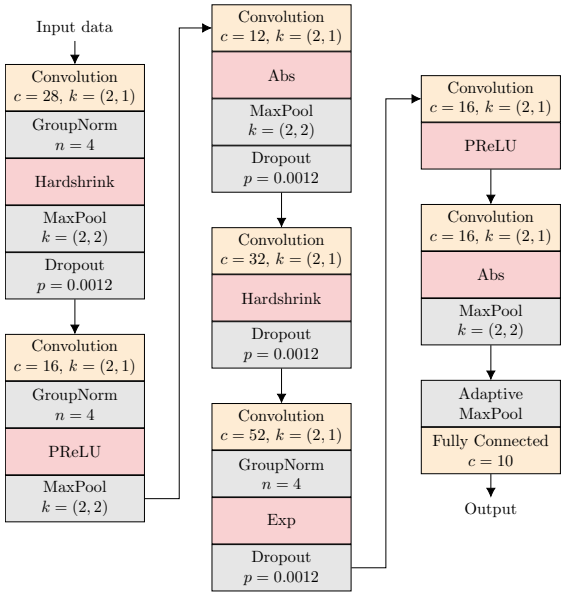


Figure 9: Optimised real-valued Architecture. c is output channels, k is kernel, n is number of groups and p is dropout rate.

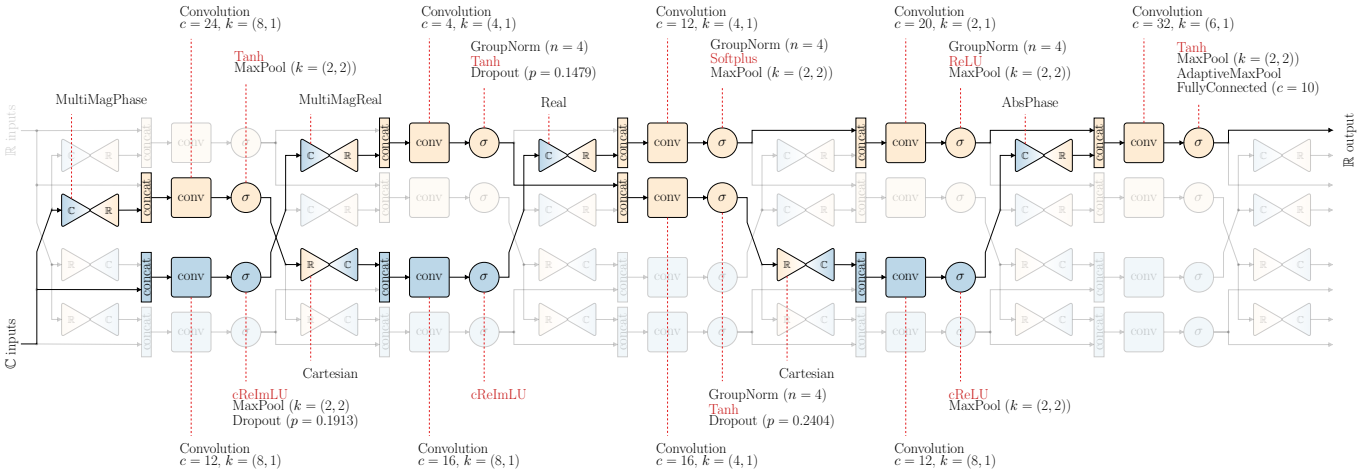


Figure 10: Optimised HNN Architecture. c is output channels, k is kernel, n is number of groups and p is dropout rate.

For the HNN, the data pre-process is the same except for one part: instead of interleaving the STFT output in the frequency dimension, we directly use the complex-valued data as input for the HNN model obtained by the method described in Section 3. Optuna is used to select the domain conversion functions, activation functions, and the usage of different layers such as max pooling and group normalisation, for real-valued paths, or batch amplitude mean normalisation (BAMN) [7], for complex-valued paths, and learning rate. As the optimisation proceeds, after the number of blocks is chosen, the parameter count is included in the framework and is progressively limited from 30k to 20k parameters. The optimised HNN architecture is shown in Fig. 10, where the domain conversion functions are described in Section 4 from equation (5) to (12) and the complex-valued activation functions are detailed in Section 5, summarised in Table 1. The results show significant improvements achieved by the HNN compared to the RVNN with a factor 3.8 reduction in cross entropy loss, a reduction from 23 to 5 incorrect classifications out of 3000, and a 9% reduction in parameter storage.

Table 2: Summary of results for the real-valued and the hybrid neural networks when applied to the AudioMNIST classification problem.

Model	Test loss	Wrong predictions	Parameters
RVNN	0.0200	23	19814
HNN	**0.0052**	**5**	**17954**

A possible explanation for the performance improvement obtained by using the HNN is that it can easily switch between domains and has access to a function repertoire difficult to emulate within a real-valued network. For future research, we intend to more closely study the passage of information through the network.

7 Discussion

We have described one method of architecture search, however, we see alternative approaches that could be used, e.g., starting with a single real-valued path, a HNN could “grow" complex-valued pathways to expand the network and become more efficient in terms of loss and parameter storage. This represents a practical option for expanding existing real-valued networks to their hybrid counterpart.

Given an optimised architecture, with specific routing, selected activation and domain conversion functions, this could indicate novel solution spaces not so easily identified within real-valued networks due to their homogeneous nature. The separation of real- and complex-valued information pathways could aid in explainability. Using targeted stimuli injected into the network, pathways and functions could be identified more easily in decoding how the network solved the problem under study.

Expanding this work to other models like variational autoencoders [11], the initial approach towards a hybrid real- and complex-valued representation could be the modification of the encoder and decoder to a hybrid counterpart – e.g., using a similar block to Figure 4. The constraint (or regularisation) of the latent space could be doubled, such that each domain contains a related Gaussian distribution at the output of the encoder. In a similar way, for a transformer structure – initially proposed by Vaswani et al. [19] – processing blocks like convolutional and fully connected layers could be expanded into a hybrid version as in Figure 4, while the multi-head attention mechanism block could be further developed into a hybrid version with information exchange between paths. Notice that a complex-valued version of the multi-head attention mechanism was already developed [16], which can serve as a baseline together with its real-valued version. In both cases, it would be naturally interesting to apply a NAS procedure such as the one proposed in Section 3, drastically reducing the amount of parameters and improving the model’s performance.

8 Conclusion

We have analysed how a RVNN processed complex-valued data and, with these insights, we have proposed a HNN architecture and a neural architecture search procedure. Comparing a RVNN and a HNN applied to a practical problem, we have demonstrated a significant performance improvement. The HNN deviates from the usual real domain solution space and importantly provides for native use of complex- and real-valued data. The novel complex-valued activation and domain conversion functions offer great flexibility in combination with the optimisation framework to explore the HNN architecture space.

As a natural expansion for the application of the HNNs, future work could consider other problems of a complex-valued nature such as audio/speech enhancement or synthesis, beamforming, biomedical-related analysis based on electrocardiogram and electroencephalogram signals, etc.

References

[1] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 2623–2631, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016.

[2] J. Bassey, L. Qian, and X. Li. A survey of complex-valued neural networks, 2021.

[3] S. Becker, J. Vielhaben, M. Ackermann, K.-R. Müller, S. Lapuschkin, and W. Samek. Audiomnist: Exploring explainable artificial intelligence for audio analysis on a simple benchmark. *Journal of the Franklin Institute*, 2023. ISSN 0016-0032.

[4] H. Du, R. P. Riddell, and X. Wang. A hybrid complex-valued neural network framework with applications to electroencephalogram (eeg). *Biomedical Signal Processing and Control*, 85:104862, 2023. ISSN 1746-8094.

[5] A. Fuchs, J. Rock, M. Toth, P. Meissner, and F. Pernkopf. Complex-valued convolutional neural networks for enhanced radar signal denoising and interference mitigation. In *2021 IEEE Radar Conference (RadarConf21)*, pages 1–6, 2021.

[6] G. Georgiou and C. Koutsougeras. Complex domain backpropagation. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 39(5):330–334, 1992.

[7] D. Hayakawa, T. Masuko, and H. Fujimura. Applying complex-valued neural networks to acoustic modeling for speech recognition. In *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1725–1731. IEEE, 2018.

[8] Y. Ju, J. Chen, S. Zhang, S. He, W. Rao, W. Zhu, Y. Wang, T. Yu, and S. Shang. Tea-pse 3.0: Tencent-ethereal-audio-lab personalized speech enhancement system for icassp 2023 dns-challenge. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–2, 2023.

[9] C. Lee, H. Hasegawa, and S. Gao. Complex-valued neural networks: A comprehensive survey. *IEEE/CAA Journal of Automatica Sinica*, 9(8): 1406–1426, 2022.

[10] S. Lv, Y. Hu, S. Zhang, and L. Xie. Dccrn+: Channel-wise subband dccrn with snr estimation for speech enhancement, 2021.

[11] L. Pinheiro Cinelli, M. Araújo Marins, E. A. Barros da Silva, and S. Lima Netto. *Variational Autoencoder*, pages 111–149. Springer International Publishing, Cham, 2021. ISBN 978-3-030-70679-1.

[12] C.-A. Popa. Deep hybrid real-complex-valued convolutional neural networks for image classification. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6, 2018.

[13] Y. Quan, Y. Chen, Y. Shao, H. Teng, Y. Xu, and H. Ji. Image denoising using complex-valued deep cnn. *Pattern Recognition*, 111:107639, 2021. ISSN 0031-3203.

[14] A. M. Sarroff. *Complex Neural Networks for Audio — digitalcommons.dartmouth.edu*. Phd thesis, Dartmouth College, May 2018. Available at https://digitalcommons.dartmouth.edu/dissertations/55/.

[15] M. Sipper. Neural networks with À la carte selection of activation functions. *SN Computer Science*, 2(6), Sep 2021.

[16] V. Tokala, E. Grinstein, M. Brookes, S. Doclo, J. Jensen, and P. A. Naylor. Binaural speech enhancement using deep complex convolutional transformer networks. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 681–685, 2024.

[17] C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. J. Pal. Deep complex networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=H1T2hmZAb.

[18] V. Tsouvalas, A. Saeed, and T. Ozcelebi. Federated self-training for data-efficient audio recognition. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 476–480, 2022.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[20] S. Welker, J. Richter, and T. Gerkmann. Speech Enhancement with Score-Based Generative Models in the Complex STFT Domain. In *Proc. Interspeech 2022*, pages 2928–2932, 2022.