Young *et al.*
Journal on Audio, Speech, and Music Processing (2026) 2026:27
https://doi.org/10.1186/s13636-026-00457-2

Journal on Audio,
Speech, and Music Processing

METHODOLOGY

Open Access

Hybrid real- and complex-valued neural network architecture

Alex Young[1*], Luan Vinícius Fiorio[2*], Bo Yang[3], Boris Karanov[4], Wim van Houtum[1] and Ronald M. Aarts[2]

Abstract

We propose a *hybrid* real- and complex-valued *neural network* (HNN) architecture, designed to combine the computational efficiency of real-valued processing with the ability to effectively handle complex-valued data. We illustrate the limitations of using a real-valued neural network (RVNN) for inherently complex-valued problems by showing how it learns to perform complex-valued convolution; learning twice as many weights as necessary. To create the HNN, we use building blocks containing both real- and complex-valued paths, where information between domains is exchanged through domain conversion functions. We also introduce novel complex-valued activation functions, with better generalisation and parameterisation efficiency. HNN-specific architecture search techniques are described to navigate the larger solution space. Experiments with the AudioMNIST dataset demonstrate that the HNN reduces cross-entropy loss and consumes fewer parameters compared to an RVNN for all cases considered. Further experiments for audio denoising also show performance gains using HNNs with a reduced model complexity when compared to its real- or complex-valued counterparts. Such results highlight the potential for the use of partially complex-valued processing in neural networks and applications of HNNs in many signal processing domains.

Keywords Architecture optimisation, Complex-valued processing, Domain conversion, Neural networks, Signal processing

1 Introduction

There is ongoing debate about the usefulness and efficiency of complex-valued processing in neural networks [1] compared to conventional real-valued architectures. Consequently, it is essential to benchmark any novel architecture against a baseline real-valued network, this methodology is used throughout this work.

The necessity for complex-valued data processing, in many fields of science where deep learning is applied, usually requires data to be converted into a real-valued representation due to limitations that neural networks (NNs) have with complex-valued parameters. For example, in audio processing, discarding the phase component simplifies data handling but reduces audio intelligibility and limits synthesis quality, as phase information is crucial for accurate sound reconstruction [2, 3]. On the other hand, [2] explored the phase estimation at short frame lengths, showing that its explicit estimation might reduce processing time. However, even when phase is retained, processing different components separately can cause phase distortion, degrading signal quality [4].

Deep complex-valued neural networks (CVNNs) have been applied recently to solving traditionally

*Correspondence:
Alex Young
alex.young@nxp.com
Luan Vinícius Fiorio
l.v.fiorio@tue.nl
[1] NXP Semiconductors, High Tech Campus 60, Eindhoven 5656 AG, Noord-Brabant, The Netherlands
[2] Department of Electrical Engineering, Eindhoven University of Technology, De Groene Loper, 19, Eindhoven 5612 AP, Noord-Brabant, The Netherlands
[3] Embedded Systems Group, Delft University of Technology, Van Mourik Broekmanweg 6, Delft 2628 XE, Zuid Holland, The Netherlands
[4] Communications Engineering Lab, Karlsruhe Institute of Technology, Hertzstr. 16, Karlsruhe 76187, Baden-Wurttemberg, Germany

complex-valued problems, e.g. radar [5] and audio processing [6] and even real-valued problems like image processing [7, 8], 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 [9]. Automatic differentiation and other overheads within CVNNs typically incur an increased training time compared to using standard functions within an RVNN [4].

The combination of real- and complex-valued processing in neural network architectures was previously considered but scarcely explored. Averaging the results of an RVNN and its complex-valued counterpart has been investigated [10] – possibly the simplest approach towards the idea of a hybrid real- and complex-valued architecture. On the other hand, [11] 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 [12], where an RVNN is followed by a complex-valued processing stage. Similarly, [13] 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 neural network design framework that systematically combines the strengths of real-valued and complex-valued processing. This framework introduces a family of complex-valued activation functions with inherent stability, an extended set of domain conversion functions and a neural architecture search (NAS) strategy for routing between real and complex domains. By integrating pathways for both types of processing, the framework enables optimal selection of activation functions and domain-specific operations, while also reducing redundancies between paths. This leads to reduced loss, parameter count and computational cost. A custom NAS is also applied to an RVNN for a fair comparison against a baseline. We also demonstrate the use of the hybrid networks' concept in a speech enhancement study case by practically hybridising a real-valued model by groups of layers, instead of a fine layer-by-layer modification. In such scenario, the design time is reduced and the use of NAS is optional for further improvements. Moreover, we confirm that the hybrid model outperforms its real- or complex-valued counterparts for speech enhancement with the same number of parameters, however, presenting a dramatic reduction of multiply-accumulate operations. Note that when referring to parameters, we count two parameters per complex weight or bias and one per real weight or bias.

The remainder of this paper is organised as follows, we first describe a toy experiment (Section 2) and highlight the insights that went on to define a general hybrid network architecture (Section 3), within which, we describe the components; architecture search (Section 4), domain conversion (Section 5) and complex-valued activation functions (Section 6). Furthermore, we describe two numerical experiments (Section 7) and sections for discussion (Section 8), conclusion (Section 9) and an appendix (Section A).

2 Toy experiment

Our initial objective was to investigate how a real-valued neural network (RVNN) processes complex-valued data when represented as pairs of real numbers corresponding to the real and imaginary components of a complex vector. To this end, we constructed a simple task involving the estimation of frequency, amplitude and phase from a short-length spectrum, including cases where the frequency is not aligned with the discrete frequency grid. In this section, we describe the dataset creation, the initial toy experiment (baseline) using an RVNN, the subsequent analysis and function identification, the development of a data flow schematic, a second experiment using an HNN and the optimisation steps leading to the third and fourth experiments.

To estimate the parameters of a complex-valued sinusoidal signal in the presence of noise, we consider the model:

$$y = ae^{i(2\pi nf/N+p)} + \nu(n), \quad n = 0, \ldots, N-1 \tag{1}$$

Where $a \in [0.1, 1.0]$ is the amplitude, $f \in [5.0, 12.0]$ is the frequency in cycles, $p \in [-\pi, \pi]$ is the phase in radians and $\nu(n)$ is complex-valued noise with magnitude in the range $\in [0, 0.01]$. With $N = 512$, the signal y is multiplied by a Hann window and converted to the frequency domain using an N point discrete Fourier transform. A training set of 500,000 samples is generated using the specified parameter ranges. From each spectrum, only the first 18 frequency bins are retained. The real and imaginary components of these bins are split and interleaved to form 36 real-valued predictors. The objective is to estimate the parameters $[f, as, ac]$, where $as = a\sin(p)$ and $ac = a\cos(p)$. Initially, we attempted to use the parameters $[f, a, p]$ directly as the model targets but observed increased error in phase estimation when the phase values wrapped around at $\pm\pi$. Using ac and as

instead eliminates the phase discontinuity issue and informs us to pay attention to phase handling in general.

We defined an RVNN model consisting of four fully connected layers[1], see Fig. 1a. The first three layers are followed by exponential linear unit (ELU) activation functions. The output sizes of the layers are 18, 14, 4 and 3, respectively. The model was trained multiple times to verify that it exhibited similar weight behaviour and provided a reference performance of 0.0220 with 1007 learnable parameters with the Adam optimiser and RMSE loss.

2.1 Weight reordering

Figure 2a shows the weights ($\mathcal{W}$) of the first fully connected layer, obtained after 2500 epochs of training and normalised from −1 to 1, where the first column on the left represents the bias. While no definitive conclusions can be drawn from this plot, it suggests that some output rows follow similar patterns. By grouping these output rows, we hypothesised that they might correspond to complex-valued multiply-accumulate (CVMAC) operations. Given the known ordering of real and imaginary input components, each output row can be interpreted as representing either the real or imaginary part of a CVMAC operation. To test this hypothesis, we defined $A_{m,n1} = \mathcal{W}r_{m,n1} + i\mathcal{W}i_{m,n1}$ for the real part and $B_{m,n2} = \mathcal{W}r_{m,n2} + i\mathcal{W}i_{m,n2}$ for the imaginary part, where m is the $\Re$ and $\Im$ input pair selection and $n1$, $n2$ are the output row selections. Combining these output selections gives $A_{m,n1} + iB_{m,n2}$ to indicate the multiplication result, conversely to find the residual, $A_{m,n1} - iB_{m,n2}$ can be used. Using a value of $1 + i$ for all layer inputs activates both real and imaginary components, allowing all weight contributions to be evaluated. To consider each multiply separately within the CVMAC, we take the magnitude, sum over the input indexes for the indication and residual and calculate the ratio avoiding division by zero:

$$fom_{n1,n2} = \frac{\sum_m |A_{m,n1} + iB_{m,n2}|}{\epsilon + \sum_m |A_{m,n1} - iB_{m,n2}|}, \quad \epsilon = 10^{-6} \tag{2}$$

By testing all combinations of pairs of output weight rows, plotting a heatmap of *fom* indicates that 5 pairs of output rows support CVMAC operations, see Fig. 2b. Arranging the pairs adjacent to each other, in order of similarity, produces region V in the top left weight plot in Fig. 3. As an example, substituting the weights from a single complex-valued multiply candidate as shown in Fig. 2c, yields the following results:

$$A_{m,1} = 0.986 + 0.819i, B_{m,3} = 0.827 - 0.986i \tag{3}$$

$$|A_{m,n1} + iB_{m,n2}| = 2.569, \quad |A_{m,n1} - iB_{m,n2}| = 0.008 \tag{4}$$

The absolute sum (indication) shows a significantly higher value than the absolute difference (residual), demonstrating the concept.

This shows that the RVNN, in part, is inherently learning to construct a complex-valued convolution using 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.

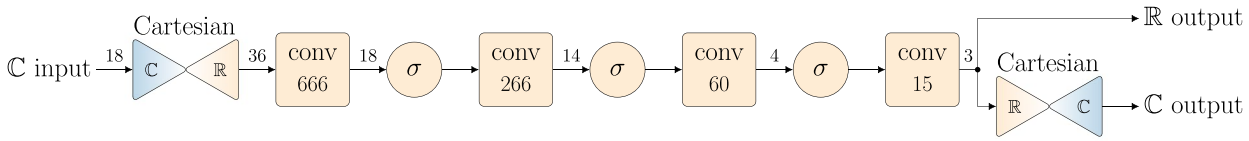
The remaining outputs of layer 1 in Fig. 3 that were not identified by the CVMAC analysis consist firstly of an unused output (4) as its input referred weights are relatively insignificant. Secondly, 7 other outputs form region U that we consider an arbitrary real-valued convolution.

Returning to region V, we observe a highly organised pattern in the weight values from reordered outputs 16 to 12. Propagating the reordering to later layers by reordering each layer's inputs associated to the previous layer's outputs, allows us to observe more potential patterns in each layer. The aim was to reorder outputs to form clusters (regions X and R), patterns (region Z) and a sparse layer from regions Q and R. A tool was constructed to manipulate layer outputs and visually check for the emergence of patterns in the current and subsequent layer weights. These separated regions allow us to better understand the flow of information in this first toy example (RVNN) and informs us to redraw the network as a schematic.

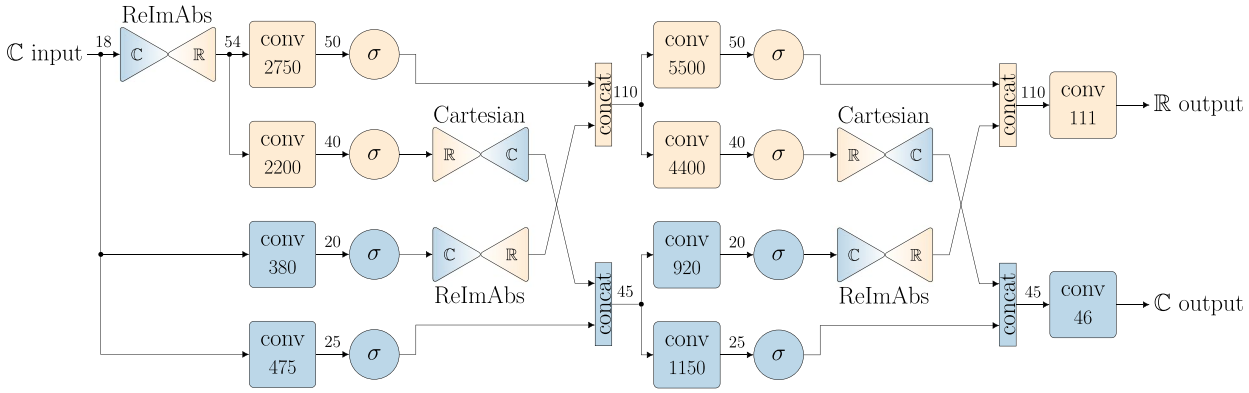
2.2 Function decoding

Figure 4 shows the identified weight regions Q to Z as small convolutions shown as square shapes and the associated information pathways, we highlight the 'complex path' to identify the signals that ultimately form the complex-valued output pair 'ac' and 'as'. The downward arrows take various points along this path to show the different variations that the signal can take, with the complex-valued output showing a 2-phase representation, 3-phase and 4-phase signals are also shown in Fig. 5. Activation functions, shown as circular shapes, indicate the type of function and the general bias level on its inputs, obtained by observing the sign of the bias from the preceding convolution. In this case, all activations have been

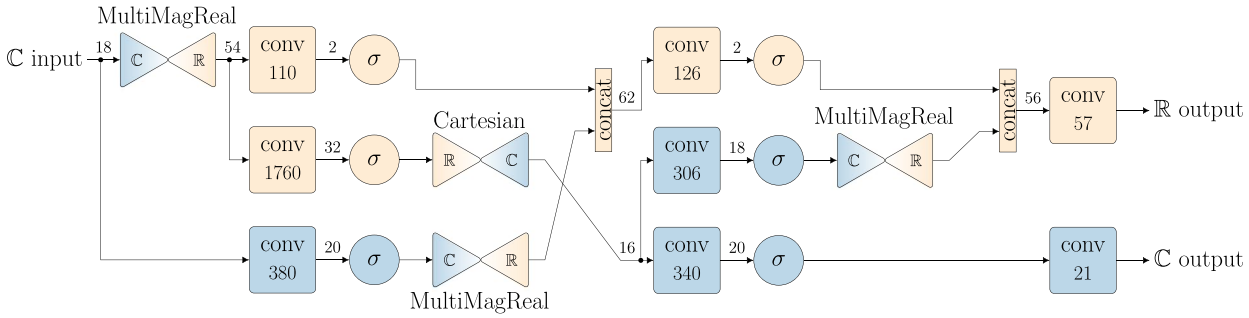
[1] In Section 2 all layers are implemented as fully connected (linear) layers. Figures show "conv" blocks only to match later sections where we generalise to convolutional implementations (a 1-D convolution with kernel length equal to input width and stride of 1 behaves identically to a linear layer)



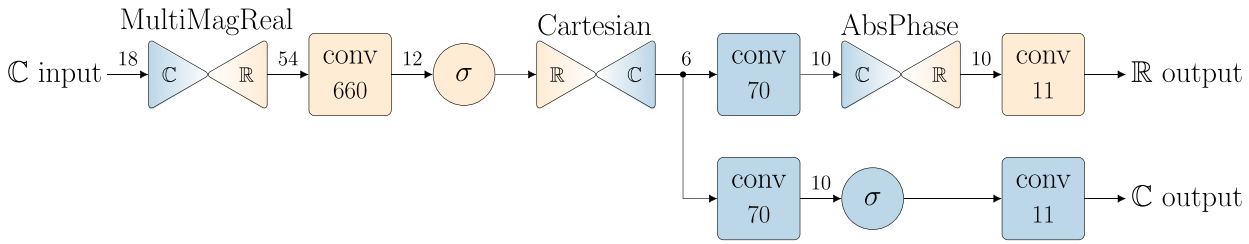
(a) First toy (RVNN): 1007 parameters, loss 0.0220 (Ref)



(b) Second toy (HNN): 20993 parameters, loss 0.0080 (Ref ÷ 2.8)



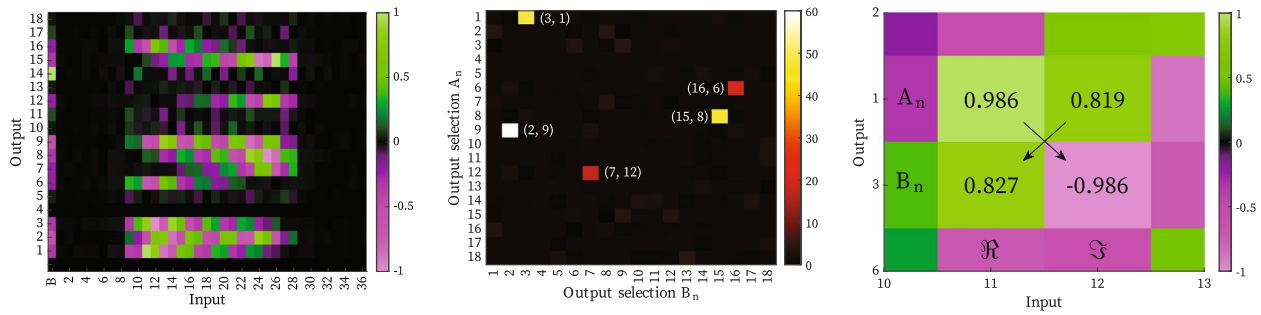
(c) Third toy (HNN): 4147 parameters, loss 0.0060 (Ref ÷ 3.7)



(d) Fourth toy (HNN): 973 parameters, loss 0.0031 (Ref ÷ 7.1)

Fig. 1 Toy example architecture evolution. All convolutional layers are implemented as fully connected 1D convolutions, where the kernel length equals the signal width and the stride is 1. The number inside each "conv" block indicates the number of parameters. Numbers on the arrows represent signal widths. The activation function σ is ELU for real-valued branches and cRecip (see Table 1) for complex-valued branches. ReImAbs (5), MultiMagReal (8h) and AbsPhase (8d) are used for complex to real domain conversion functions. For real- to complex-valued domain conversions, the conventional Cartesian (7e) function is used. (Ref) refers to the loss of the RVNN in Fig. 1a, with subsequent ÷ indicating the proportion of loss reduction

selected to be ELU. The bias sign can be inferred from the weight maps with the label 'B', for example in Fig. 3, the first column adjacent to the V region is purple indicating a negative bias so the following ELU activation function will have a negative bias moving a zero-mean input signal into the non-linear region.



(a) Trained weights (b) Complex MAC detection (c) Reordered weights detail

Fig. 2 Weight reordering analysis and result. Plots **a**–**c** illustrate key aspects of the first toy experiment. **a** shows the trained weights and biases of layer 1. **b** presents a heatmap of complex-valued multiply-accumulate detection results, where each coloured coordinate represents a combination of layer 1 outputs forming a complex-valued pair. **c** provides a detailed view of reordered weights from output channels 1 and 3, corresponding to one set of real and imaginary inputs. The weight values reveal that the network has learned an almost perfect conjugate relationship

Using the trained network for inference on generated datasets with fixed amplitude and frequency parameters, we varied the phase parameter continuously from $-\pi$ to $+\pi$. This allowed us to observe how a rotating phasor at the output is internally represented by the network. We found that 3- and 4-phase activations with positive values were present as the information was transformed into phasor outputs. These observations provided the first insights into how a complex-value could be mapped into a multi-dimensional positive real space and vice-versa. Based on this behaviour, we developed a novel domain conversion function, 'MultiMagReal', for the third toy experiment, see (8h) and Fig. 1c.

The schematic shown in Fig. 4 shows a separation into real- and complex-valued pathways and also cross connections between the domains, blocks W, X, Y and Z can be interpreted to process $\mathbb{R} \to \mathbb{R}$, $\mathbb{C} \to \mathbb{R}$, $\mathbb{R} \to \mathbb{C}$ and $\mathbb{C} \to \mathbb{C}$ information respectively. Building on this concept of direct and cross-domain information exchange, we designed a second toy experiment (see Fig. 1b) using our first HNN, using the same data set as the first toy experiment. This experiment also introduced the idea of testing different domain conversion functions. For all complex- to real-valued conversions the ReImAbs function was used, see (5) and for real- to complex-valued conversions the Cartesian function was used, see (7e).

$$\text{ReImAbs: } \{y_1 = \Re(z),\ y_2 = \Im(z),\ y_3 = |z|\}. \tag{5}$$

We selected a simple complex-valued activation function, cRecip, see (1). All convolutions were made fully connected and dimensioned so that the network was not limited by constraints in signal width. The parameter count, at this stage was just under 21 k parameters (complex-valued weights and biases counted twice), however, we noticed a significant reduction in loss by a factor of 2.8 compared to the first toy experiment. To improve performance, a third toy experiment (HNN) was designed. We noticed in the previous experiment that the $\mathbb{C} \to \mathbb{C}$ pathway on the input (left) side, 'conv 475' had weights that had a maximum magnitude of $< 10^{-20}$, so that pathway was removed. Further reductions were made with speculative culling of signal widths through the network, reducing parameter count to 4147. Changing domain conversion functions from ReImAbs to MultiMagReal (8h) helped to reduce loss by a factor of 3.7 compared to the first toy experiment. This also confirmed the usefulness of being able to easily swap conversion functions. The fourth toy experiment (HNN) was designed to have less parameters than the RVNN baseline. Further speculative culling reduced the parameter count, then we changed the domain conversion function on the right hand side from MultiMagReal to MultiMagPhase (see (8i)) and noticed a reduction in loss. Realising that the combination of the cRecip activation, $Z_o = \frac{z}{|z|+\epsilon}$ with $\epsilon = 10^{-5}$ and MultiMagPhase (one phase), $y_0 = |z||\angle z|/\pi$ could be replaced by a simpler function, we designed another domain conversion function, AbsPhase:

$$Z_o = \frac{z}{|z|}, y_0 = \frac{|Z_o||\angle Z_o|}{\pi} = \frac{|\angle z|}{\pi} \tag{6}$$

These changes brought the parameter count slightly lower than the first toy experiment from 1007 to 973 and reduced the loss by a factor of 7.1, illustrating the potential of the HNN architecture. The problem with this approach is the time spent on manual tuning of hyperparameters and dimensions.

Allowing a general purpose network to have paths of complex-valued processing while still maintaining real-valued branches would be a natural solution to facilitate

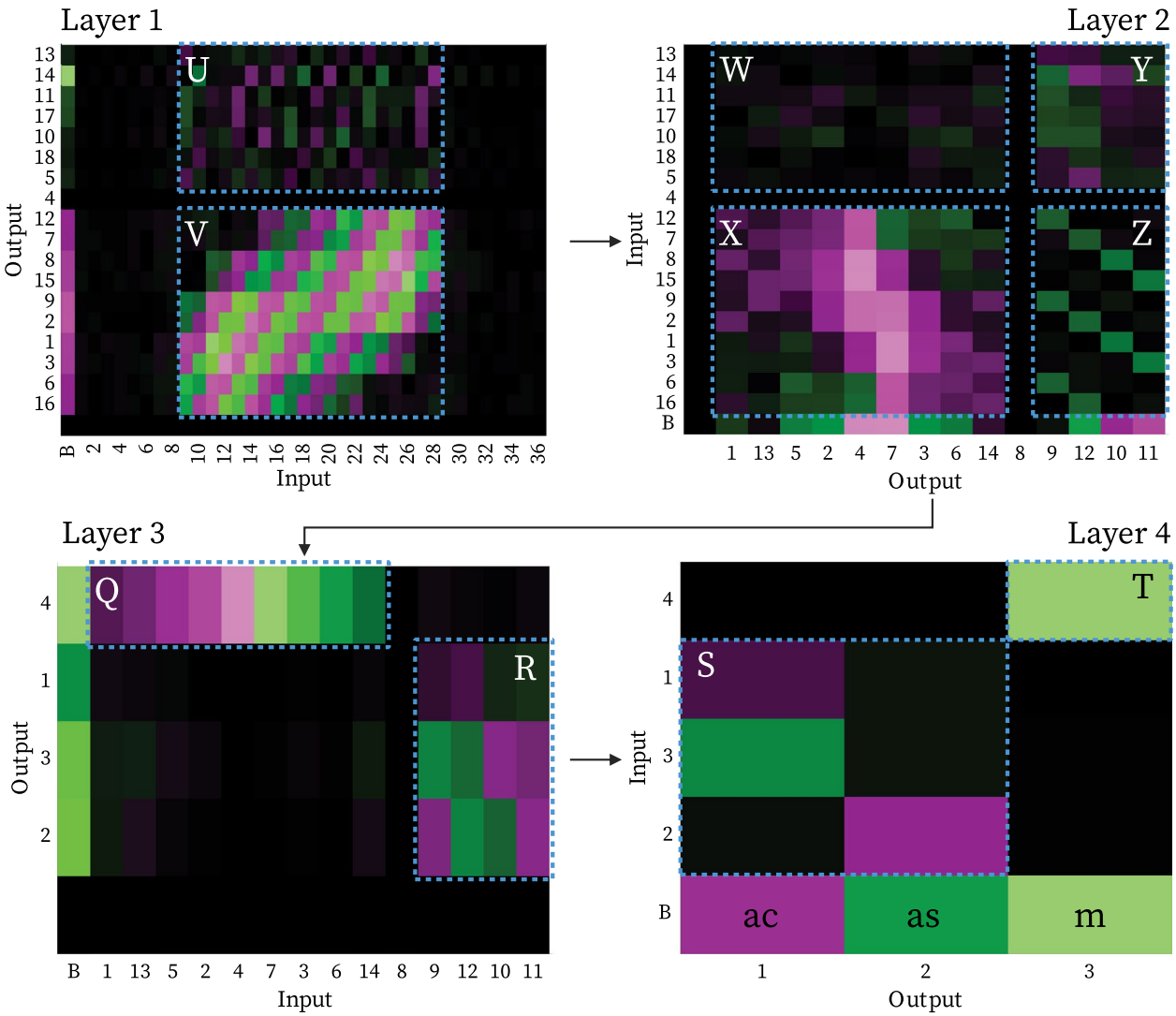


Fig. 3 Plot of reordered and normalised weights for each layer's convolutions for the first toy experiment (RVNN). Weights for layer 1 are shown top left, progressing to weights for layer 4 bottom right. The arrows indicate how the output of each layer is followed by the input of another. Distinct regions in the weight patterns are highlighted with dashed blue lines, presenting a letter as its label. Refer to the colour bar in Fig. 2a

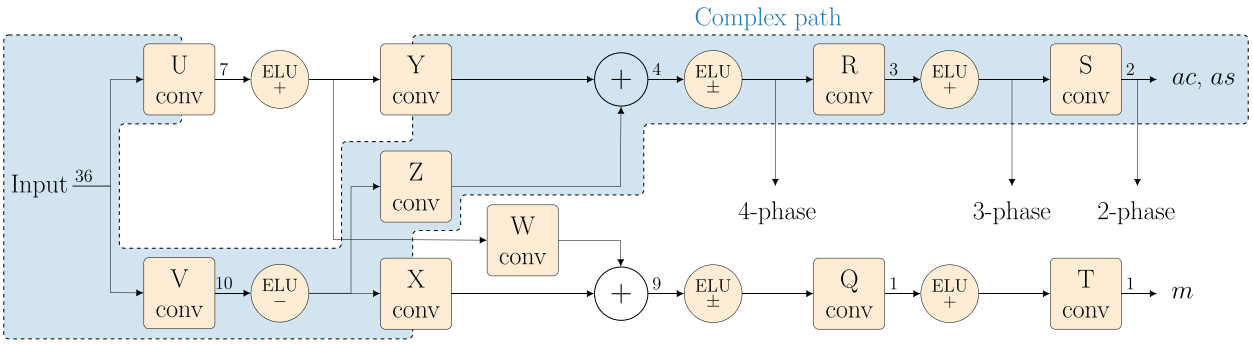
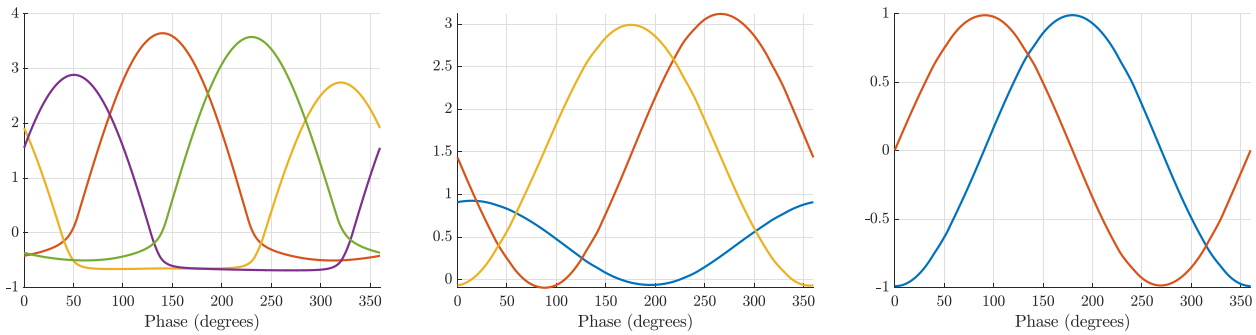


Fig. 4 Detailed schematic of the trained and decoded first toy experiment (RVNN), given weight regions defined in Fig. 3. The complex path shows the activations that represent the path where complex-valued data is conveyed in various forms, learned by the RVNN. When a number is indicated after the input, a layer, or a summation, it indicates size in activation signals



(a) 4-phase (b) 3-phase (c) 2-phase

Fig. 5 Multi-phase real-valued functions observed in the 4-, 3- and 2-phase indication in Fig. 4

complex-valued processing and may offer advantages for real-valued processing. 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.

3 Architecture

The HNN is built upon functional blocks 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 5. 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 Fig. 6, 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 ($\mathbb{C} \to \mathbb{R}$ and $\mathbb{R} \to \mathbb{C}$) 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 as described in Section 4. Multiple building blocks can be combined in series and/or in parallel. In this work, we specifically try to build a network by connecting such blocks in series.

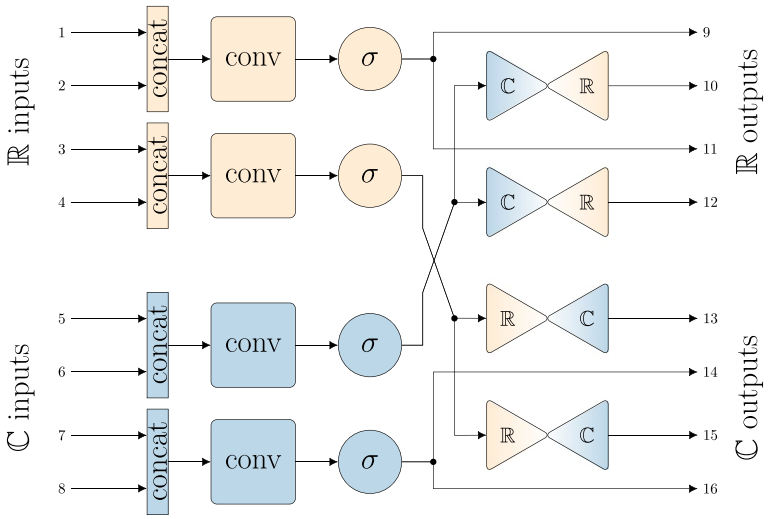


Fig. 6 HNN functional block

If the system input data is only real- or only complex-valued, then the unused ports can be either removed—with the associated routing in the next block—or domain conversions can be performed from the used input to the other domain input. Similarly, if the system outputs are not required to be both real- and complex-valued, the unused output can be removed together with all dependent routing.

Furthermore, the initial step in designing an 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 Fig. 6, 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 short-time Fourier transform (STFT) of a signal, the complex-valued inputs 5 and 7 or 6 and 8 from Fig. 6 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. Similarly, the same can be done for the outputs. Considering, e.g. a real-valued output, a backward dependency check can be done to remove unnecessary connections.

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.

4 Architecture search

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 [14], originally developed in 2019, is a hyperparameter optimisation framework that returns an optimised neural network configuration by evaluating the performance of different architectures and hyperparameters. Thus, we adopted a framework for building HNNs using Optuna. The methodology comprises eight sequential phases aimed at tailoring and refining an HNN to suit a specific task. The sequence of phases is shown in Fig. 7 and is described next.

Initially, the input and output pathways of the network are adjusted for the needs of the task and their dependencies are checked to mitigate redundancy ("Customisation" phase). The number of blocks, or depth of the network ("Depth search" phase), is selected based on performance, where balancing model complexity and computational efficiency is important. Until the depth search, 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. Additionally, a preliminary learning rate must be established, serving as a starting point.

Whenever the architecture is changed by the NAS, the "Routing update" phase identifies unnecessary paths in all blocks, which have then their dependencies adjusted to reduce redundancy ("Dependency check" phase), in an iterative process. After these steps, the architecture is refined in the "Channel search" phase, involving the optimisation of blocks within the architecture, including adjustments in convolution channel number and whether other (optional) function layers should be included within each block.

The activation functions and domain conversion functions are chosen in a later phase, named "Activation & DC choice". 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. The domain conversion functions are selected from a candidate list, which serve to interchange information between real- and complex-valued pathways.

Moreover, a "Channel refinement" phase aims to refine blocks within the architecture, including adjustments in convolution channel number and whether other function layers should be included within each block. Finally, the optimisation of the learning rate and the selection of a better optimisation algorithm ("Hyperparameters

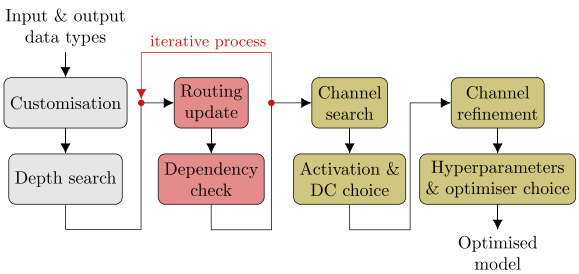


Fig. 7 HNN architecture search process

and optimiser choice" phase) is performed, aimed at enhancing convergence speed and overall efficiency. This methodology facilitates the comprehensive building and optimisation of HNN architectures tailored to a specific task.

5 Domain conversion

HNNs employ conversion between the real- and complex-domains, this is usually limited to Cartesian or polar conversions. In the RVNN 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 an 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.

We strive to define diverse forms of conversion from the real- to complex-domain, mostly based on Cartesian and polar functions. Using between one to three real-valued inputs x_1, x_2 and x_3 to produce a complex output z, we define seven domain conversion functions.

Single input real- to complex-valued functions (expansion):

$$\text{Real: } z = x_1; \tag{7a}$$

$$\text{Exp: } z = e^{i\pi x_1}; \tag{7b}$$

$$\text{Sqrt: } z = \sqrt{x_1}; \tag{7c}$$

$$\text{MagExp: } z = |x_1|e^{i\pi x_1}; \tag{7d}$$

Dual input real- to complex-valued functions (no loss of information):

$$\text{Cartesian: } z = x_1 + ix_2; \tag{7e}$$

$$\text{Polar: } z = x_1 e^{i\pi x_2}; \tag{7f}$$

Triple input real- to complex-valued function (contraction):

$$\text{Rotation: } z = (x_1 + ix_2)e^{i\pi x_3}; \tag{7g}$$

We define a complementary set of complex- to real-domain conversion functions producing one to three outputs (y_1, y_2 and y_3) from a complex-valued input z, including improved handling of phase information. Using raw phase information can cause problems due to the discontinuity of the argument function as the phase value wraps around $\pm\pi$, a circular loss function may help mitigate the issue if phase estimation is needed as a target. Alternatively, using the absolute value of the phase eliminates the discontinuity. Nine functions are offered where information is lost (lossy) or where no information is lost (lossless). ($\angle$ represents the argument or angle function).

Lossy functions:

$$\text{Real: } y_1 = \Re(z); \tag{8a}$$

$$\text{Mag: } y_1 = |z|; \tag{8b}$$

$$\text{SquareMag: } y_1 = |z|^2; \tag{8c}$$

$$\text{AbsPhase: } y_1 = |\angle(z)|/\pi; \tag{8d}$$

$$\text{MagAbsPhase: } \{y_1 = |z|,\ y_2 = |\angle(z)|/\pi\}; \tag{8e}$$

Lossless functions:

$$\text{Cartesian: } \{y_1 = \Re(z),\ y_2 = \Im(z)\}; \tag{8f}$$

$$\text{Polar: } \{y_1 = |z|,\ y_2 = \angle(z)/\pi\}; \tag{8g}$$

Multi-phase functions (lossless when $N_p \geq 3$):

$$\text{MultiMagReal: } y_n = (|z| + \Re(ze^{-2i\pi n/N_p})/2, n = 0...(N_p - 1); \tag{8h}$$

$$\text{MultiMagPhase: } y_n = (|z||\angle(ze^{-2i\pi n/N})|)/\pi, n = 0...(N_p - 1); \tag{8i}$$

The multi-phase functions were inspired by observations in Section 3, to reproduce the positive biased functions ($y_n \geq 0$), a combination of the magnitude (providing positive offset) and a phase-shifted version of the complex-valued input was used in the first case. Similarly, we define an angle-based conversion function that generates outputs that have a linear phase characteristic. By equally spacing the phase shifts, any number of outputs can be produced. N_p is the number of output phases, $N_p = 3$ is a fair choice, reducing the preliminary network architecture search space and maintaining a conversion without loss of information.

6 Complex-valued activation functions

Many complex-valued activation functions have been proposed [9, 15]. Starting from the function $Z_o = z/(|z| + \epsilon)$ [16], 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 indication that similar functions in this class can be used effectively with both real- and complex-valued arguments. See Appendix A for a more detailed description. The following proposed functions are further generalisations:

$$Z_o = \alpha z + \frac{1}{|z|^q + \epsilon} \sum_{n=0}^{p} k_n z^n; \tag{9a}$$

$$Z_o = \alpha z + \frac{1}{\sqrt{|z|^q + \epsilon}} \sum_{n=0}^{p} k_n z^n, \tag{9b}$$

where p and q are integers, k_n is real- or complex-valued and $\epsilon > 0$.

The activation function (9a) includes approximations or substitutes for the rectified linear unit (ReLU), Tanhshrink and absolute real-valued activation functions. Function (9b) includes 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.

We have also defined two purely complex-valued activation functions to add diversity to the function library that modulate the magnitude of the input depending upon the input phase:

$$Z_o = z(1 - |\alpha| + \alpha \text{sign}(v)), \tag{10}$$

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

$$Z_o = z\left(1 - |\alpha| + \frac{\alpha v^q}{|z|^q + \epsilon}\right), \tag{11}$$

which has a smooth transition instead, where $v = \max(0, \Re(z) - k_1|\Im(z)| - k_0)$.

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, Softplus and Abs (an uncommon activation function, identified in [17]); and two cases for (10) and (11). The parameters q, k_n, α and ϵ, used to achieve the aforementioned shapes, are shown in Table 1 with their proposed names. The parameters of the chosen activations can be tuned in later optimisation processes if desired. This section is supported by Appendix A, where each complex-valued activation function is detailed.

7 Numerical experiments

In this section, we validate the proposed neural network architecture from two different perspectives. At first, we explore its capabilities for audio classification of clean speech, where the data are clean and we are able to more systematically understand the behaviour of the different real- and complex-valued branches of the network. After that, the architecture is tested for a more challenging application—speech denoising. In the latter, we apply the concept of hybrid processing more generally in an architecture that is built upon audio processing prior knowledge.

7.1 Speech classification

To validate the practicality and advantages of the HNN model, we employed the AudioMNIST dataset [18] as a proof of concept. This dataset comprises 30,000 audio samples of spoken digits (0–9) produced by 60 distinct speakers. To evaluate the HNN, we compare it to an optimised RVNN model using a NAS process similar to what is described in Section 4. The primary objective is to evaluate the performance of both the HNN and RVNN models in the classification of spoken digits.

7.1.1 Data

The raw audio data from AudioMNIST at 48 k samples/s rate is normalised (so its root-mean-square value equals 1) and zero-padded to 1 s of duration, where the audio content is time-shifted with a random offset for each access to satisfy time invariance requirements. White Gaussian noise is then added to the padded audio signals, with a chosen signal-to-noise ratio (SNR). An STFT of 960 points and 50% overlap using a Hann window is applied to the signal providing a time and frequency grid resolution of 10 ms and 50 Hz, respectively. For the

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

Name	Eq.	q	α	k_0	k_1	k_2	k_3	ϵ
cRecip	(9a)	1	0	0	1	0	0	0.01
cReLU	(9a)	1	0.5	0	0	0.5	0	0.01
cAbs	(9a)	1	0	0	0	1	0	0.01
cTanhshrink	(9a)	2	0	0	0	0	1	2
cTanh	(9b)	2	0	0	1	0	0	1
cSoftplus	(9b)	2	0.5	2.134	0	0.5	0	9.481
cReImLU	(10)	–	0.95	0.1	1	–	–	–
cRecipMax	(11)	2	0.9	0.1	0.5	–	–	0.1

RVNN, 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 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. For the HNN, the data processing procedure remains largely consistent, with one notable exception: instead of interleaving the real and imaginary components of the STFT output along the frequency dimension, we utilise the complex-valued data directly as input for the HNN model.

7.1.2 NAS

As a baseline, we consider the RVNN model from [19], simplified to have convolutions operating in the frequency dimension of the input data. For a fair comparison, we applied a similar optimisation procedure as described in Section 4. We define, for the RVNN, three phases of optimisation: (i) customisation and block number—the optimal configuration within each block and the number of blocks is selected, where a block in the real-valued model is composed by a convolution, an activation function and optional processing functions (pooling, normalisation and dropout); (ii) activation choice—the activation functions for each block are selected; (iii) hyperparameters and optimiser choice—after finalising the structure of the RVNN, the hyperparameters, such as the learning rate, are optimised, as well as the optimiser is chosen to facilitate/enhance convergence. The considered activation functions in the real-valued architecture search procedure were: ReLU, Softplus, Tanh, Abs and Tanhshrink.

In [17], the authors employed 48 activation functions and ranked their performance, resulting in an extensive search space. To reduce the optimisation cost, we categorised these activation functions into five groups based on their shapes and selected the top-performing functions from each category in our experiments.

Additionally, to optimise the balance between memory usage and performance, in the structure refinement phase we employ a single-objective search methodology within Optuna, regarding the minimisation of the validation loss, but set constraints on the minimum and maximum number of parameters for potential architectures. This approach enables the exclusion of models which either under-perform or have an excessive number of parameters.

The architecture of the HNN is determined as described in Section 4. For the optional function of normalisation, batch amplitude mean normalisation (BAMN) [20] is considered for complex-valued paths. The real-valued activation functions in the HNN are consistent with those employed in the RVNN, while the domain conversion and complex-valued activation functions are introduced in Sections 5 and 6, respectively.

The initial models when NAS starts have all viable connections made, so the parameter count and loss are very high, for example, 184,714 parameters with a loss of 2.6 was observed.

To conserve space, we present only the architectural diagram and analysis processes corresponding to the −5 dB noise level, which represents the most challenging scenario considered and is shown in Fig. 8 for the RVNN and Fig. 9 for the HNN. The RVNN models for all noise levels are the same architecturally, differing only in activation functions, channel width and kernel size. The HNN models for no noise and 0 dB noise have similar architectures, differing from the −5 dB model by only having 1 complex-valued pathway in the same position towards the output.

7.1.3 Performance

Table 2 and associated Fig. 10 present the test loss and number of parameters for the RVNN and HNN, evaluated with no noise, 0 dB and −5 dB SNR noise. For the

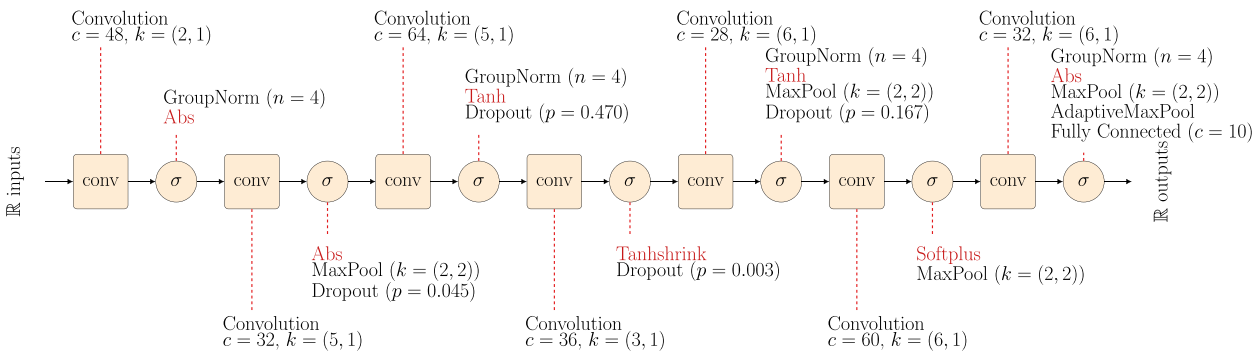


Fig. 8 Optimised RVNN model, where c is output channels, k is kernel size, n is number of groups and p is dropout rate

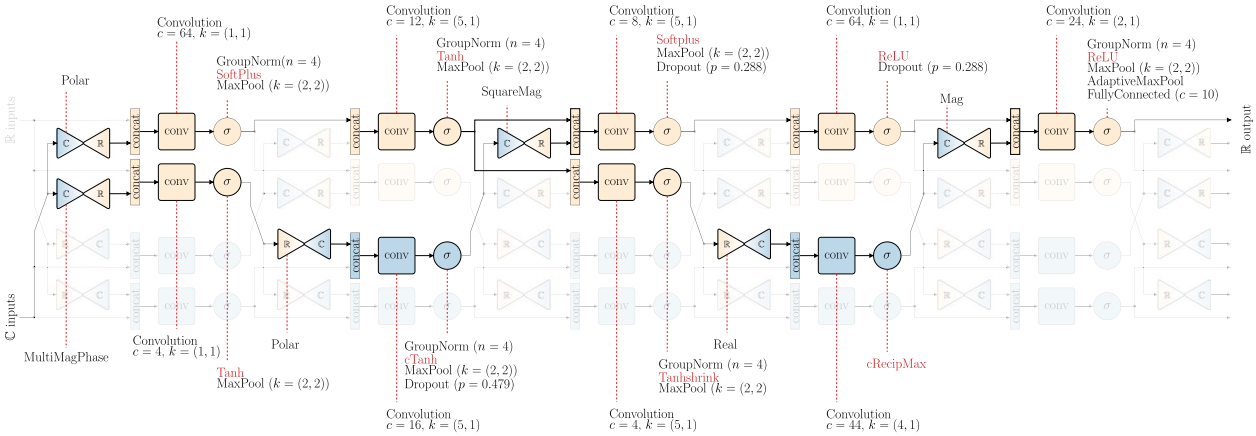


Fig. 9 Optimised HNN architecture at −5 dB SNR, where *c* is output channels, *k* is kernel size, *n* is number of groups and *p* is dropout rate

Table 2 Comparison of RVNN and HNN performance under different noise conditions

SNR	Model	Test loss	Parameters
No noise	RVNN	0.0066	39 k
	HNN	**0.0052**	**18 *k***
0 dB	RVNN	0.0329	41 k
	HNN	**0.0097**	**34 *k***
−5 dB	RVNN	0.1150	53 k
	HNN	**0.0393**	**38 *k***

bold indicates best results in each case

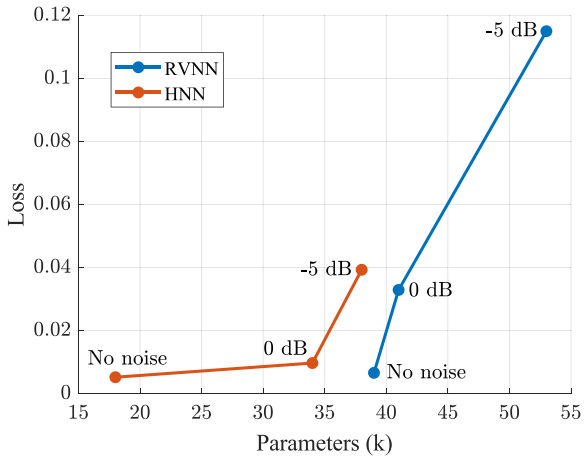


Fig. 10 Performance and loss for RVNN and HNN models trained for no noise, 0 dB SNR and −5 dB SNR

noiseless case, the HNN could perform slightly better, in terms of test loss, than the RVNN, with a reduction of parameters of 54%. Notably, in the −5 dB SNR scenario, where the noise level exceeds the signal strength, the number of parameters in the RVNN increases significantly. Despite this, the HNN is able to achieve much better performance with fewer parameters compared to the RVNN. Furthermore, the performance of the RVNN degrades more rapidly with an increase in noise level, indicating HNN robustness in the presence of noise.

To evaluate the robustness of RVNN and HNN models against noise, we conducted inference only tests at different noise levels covering −10 dB to +10 dB SNR, see Fig. 11. This study gives some insight into generalisation of the classification task as the models were only trained with no noise, 0 dB SNR or −5 dB SNR. Models trained with no additive noise perform poorly in both RVNN and HNN cases, even at +10 dB SNR, performance was degraded, this implies that noise augmentation should always be applied during training. While the −5 dB models generalise better at very low SNR levels, from −2 dB to +10 dB they perform worse than the 0 dB models with smaller margins for the HNN model. Both 0 dB models show the best generalisation, the HNN model has best performance from −5 dB to +10 dB so could be the best for practical use.

A possible explanation for the performance improvement obtained by using HNNs is that they effectively integrate real-valued and complex-valued processing paths, leveraging the advantages inherent in both domains. The complex-valued paths are capable of capturing more complicated phase relationships within the data, which can be highly beneficial for tasks involving signal processing or time series analysis. Due to the integration of complex-valued paths and the utilisation of domain conversion functions, the network may achieve more efficient parameter representation. This efficiency is particularly crucial in scenarios with limited computational resources or when attempting to mitigate overfitting. The freedom of using various layers,

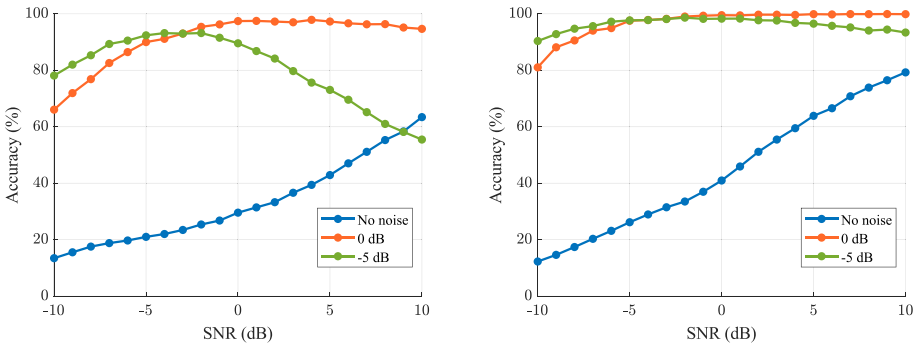


Fig. 11 Performance over SNR for RVNN and HNN models trained at specific noise levels indicated in the legend

Table 3 AudioMNIST classification accuracy and number of parameters for the proposed RVNN, HNN and other baseline approaches

Method	Accuracy (%)	Parameters
HNN (no noise, ours)	98	**18 *k***
RVNN (no noise, ours)	98	39 k
CNN [21]	98	500 k to 1 M
CNN [18]	97	∼1 M
CNN [22]	97	∼800 k
AlexNet [23]	96	∼2 M
AudioNet [23]	93	∼500 k

The baseline values were taken from each reference

bold indicates best result

normalisation techniques and activation functions provides a high degree of architectural flexibility. The network is designed through an extensive and systematic NAS procedure, as proposed in Section 4, to incorporate both real- and complex-valued processing paths, making it both robust and efficient, offering significant advantages in areas where traditional real-valued neural networks may be insufficient.

7.1.4 Comparison to other approaches

For the sake of comparison, we show the classification accuracy and total number of parameters for NN-based methods developed for the AudioMNIST classification task without added noise in Table 3. For the proposed hybrid architecture (HNN), the number of parameters is drastically reduced when compared to the baselines and our reference RVNN. Tailored NAS procedures are applied to both RVNN and HNN models, yielding drastic reductions in parameter count whilst maintaining high accuracy, moreover, the HNN parameter count is less than half of the RVNN count, this showcases the gains that can be achieved when a hybrid approach with both real- and complex-valued layers is employed.

7.1.5 Activation function interpretation

As an additional analysis, we aim to interpret the choice of activation functions from the NAS process, first looking at the RVNN model. Table 4 shows the top 5 (best performing at the top, ranked by performance, 1–7 represent the building block number, with block 1 closest to the input) activation combinations for a seven-block RVNN model. From this table we can find that in the first two blocks, the Abs activation function is selected for the top three combinations. This suggests that the absolute value operation might be particularly effective in the initial stages of the network. The Abs function applied to the separate real- and imaginary-parts of the complex-valued input likely aids in extracting the magnitude of each complex-valued input, without completely destroying the phase content, folding 360° inputs to 90°, which

Table 4 Top 5 activation function combinations selected per block by Optuna for the RVNN at −5 dB SNR

1	2	3	4	5	6	7	Loss
Abs	Abs	Tanh	Tanhshrink	Tanh	Softplus	Abs	0.2054
Abs	Abs	Tanh	Tanhshrink	Tanh	Softplus	ReLU	0.2202
Abs	Abs	Tanh	Tanhshrink	Tanh	Softplus	Abs	0.2223
ReLU	Tanh	ReLU	Tanhshrink	ReLU	Abs	Abs	0.2228
ReLU	Tanh	Tanh	Tanhshrink	Abs	Softplus	Tanhshrink	0.2289

might simplify the learning process in the subsequent layers[2].

In the later blocks "Tanh" (saturation) and "Tanhshrink" (thresholding) activation functions are consistently used, these can be seen as complementary functions. This combination may offer a balanced approach to managing signal amplitude while maintaining non-linearity, helping in the refinement of feature maps. While ReLU is well-known for its simplicity and efficiency, its selective use suggests that it may only be effective when combined with other functions like Abs and Tanh in this specific case. Softplus, a smooth approximation of ReLU, appears to be preferred in deeper layers, possibly due to its ability to support stable learning and smoother gradient propagation.

Importantly, the variation in loss over the top 5 results is relatively small at around 11%, this is within the expected variance in loss when retraining this model. In the HNN case, interpretation can be aided by observing the separation of information into distinct pathways and examining the function choices within these partial networks as above. Interestingly, the formation of similar pathways over multiple NAS sessions indicates a recurring solution pattern.

7.2 Speech denoising

For speech denoising, we follow the experiments and settings from [24], where clean speech is combined with environment noise files. The LibriTTS dataset [25] is used for clean speech data and the TAU2019 dataset [26] for environmental noise. We then define the experiment objective: compare the performance of a real-valued convolutional neural network to its complex-valued and hybrid counterparts.

Unlike the speech classification experiment, in this section we try to expand the concept of hybrid networks by considering the functional block in a different, *macro* perspective. Instead of constructing a building block from individual convolutional layers, as shown in Fig. 6, we now treat entire encoder–decoder structures as the functional unit, replacing single convolutions and activation functions. In other words, the whole model is regarded as a single building block. This format presents advantages from a practical perspective, since it does not result in a combination of building blocks that would require further architecture search. Next, we describe the data augmentation process.

[2] This inspired us to try to estimate complex magnitude using only Abs functions: $|z| \approx a(u+v) + b|u-v|$, where $u = |\Re(z)|$, $v = |\Im(z)|$, $a = 0.6791$ and $b = 0.2813$, giving a maximum error of 3.96%.

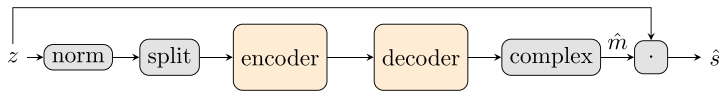
7.2.1 Data

The raw audio from both LibriTTS, sampled at 24 k samples/s and TAU2019, sampled at 48k samples/s, is converted to 16 k samples/s. For each noise file (fixed 10 s of duration), random clean speech files are selected and concatenated. Fade-in and fade-out are applied to all audio files with a random (uniformly distributed) duration from 0.2 to 0.3 s. The fade curves are calculated as $0.001e^{6.908 \cdot t}$, allowing for a dynamic range of 60 dB. Both the 10-s noise file and the (concatenated) clean speech files are combined at a random (uniform) SNR value from -5 to 20 dB. The combination is converted to the time-frequency domain via STFT, with a Hann window of 256 samples long, a Fourier transform of the same length and 50% overlap between windows. Then magnitude normalisation is applied to the complex STFT z as $z_{mag} = 20\log_{10}(|z| + 1e^{-8})$ and scaled to the range [0,1] corresponding to -80 to 0 dB. Each augmented data file consists of 10 s of audio.

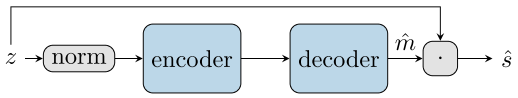
For training, 100 h of raw clean speech are combined with 40 h of raw noise files, from the LibriTTS train-clean-100 subset and the entire development set of the TAU2019 dataset, for a total of 100 h of training data. For evaluation, the LibriTTS test-clean subset (approximately 8.6 h of data) is used in combination with the TAU2019 evaluation set (20 h of data), resulting in 20 h of augmented evaluation data.

7.2.2 Architectures

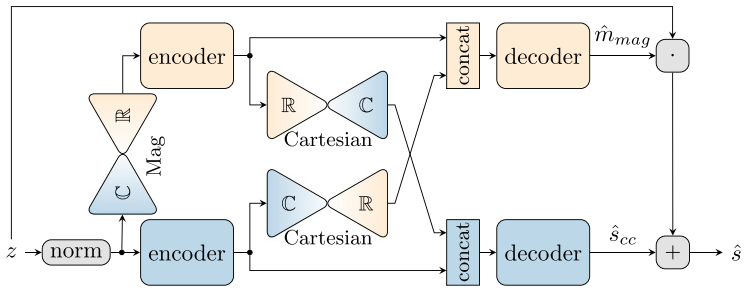
We consider a real-valued convolutional denoising autoencoder (rCDAE) from [24], which is a fully convolutional, real-valued neural network architecture. The rCDAE takes as input the concatenation of real and imaginary parts of the normalized noisy-speech STFT z, estimating the real and imaginary parts of a complex mask $\hat{m}$ [27]. The convolutions only operate in the frequency dimension, providing a low-complexity approach to speech enhancement. By changing the output size of each layer, we define a complex CDAE (cCDAE) with an equivalent number of parameters. In the complex case, however, the input is the complex STFT z and the output is the estimated complex mask $\hat{m}$. To arrive at a hybrid network equivalent to the CDAE (hCDAE), we first define two branches, a real- and a complex-valued branch. Then, we interconnect them at their bottleneck for information exchange, following the initial concept from Fig. 6. These branches adopt the same overall architecture as the rCDAE and cCDAE. However, the output sizes of each encoder/decoder layer are adjusted so that the combined parameters of the hCDAE's encoder/decoder match those of the corresponding rCDAE and cCDAE components. The architectures and layer sizes can be seen in Fig. 12 and Tables 5 and 6.



(a) Real-valued CDAE. In this diagram, the complex split function is represented by $\text{split}(z) = [\mathcal{R}(z),\ \mathcal{I}(z)]$ and $\text{complex}([\mathcal{R}(\hat{m}),\ \mathcal{I}(\hat{m})])$ outputs a complex mask $\hat{m}$



(b) Complex-valued CDAE



(c) Hybrid CDAE. The used conversion functions are indicated in the diagram

Fig. 12 Schematic of the real-, complex- and hybrid-valued CDAE. Each encoder is composed of four convolutional layers, where each layer is followed by a ReLU (real case) or cReLU (complex case), except the last, with a Tanh (real case) or cTanh (complex case). The decoder has four transposed convolution layers. Complex layers are defined as aforementioned in the manuscript. The domain conversion layers of the hCDAE are both Cartesian, following Eqs. (7e) and (8f). All convolution kernels are (8,1), with stride (2,1) and padding (2,1). Yellow indicates real-valued and blue indicates complex-valued. The layer sizes for each model are shown in Tables 5 and 6. The hat symbol (^) stands for estimated and the "norm" block normalizes the input as described in Section 7.2.1

Table 5 Encoder layers output size adopted for the denoising models shown in Fig. 12

Model	1	2	3	4	Parameters
rCDAE	16	32	64	128	86 k
cCDAE	16	18	44	96	86 k
hCDAE real branch	16	18	44	96	43 k
hCDAE complex branch	8	16	32	64	43 k

Rows with numbers indicate the layer's index. Each complex value is counted as two real-valued parameters

Table 6 Decoder layers output size adopted for the denoising models shown in Fig. 12

Model	1	2	3	4	Parameters
rCDAE	64	32	16	1	86 k
cCDAE	44	18	16	1	86 k
hCDAE real branch	22	14	8	1	43 k
hCDAE complex branch	20	14	8	1	43 k

Rows with numbers indicate the layer's index. Each complex value is counted as two real-valued parameters

Importantly, Table 7 shows the multiply-accumulate operations (MACs) for the considered models. Note that the purely real-valued and complex-valued models (rCDAE and cCDAE) have a similar number of MACs, with the difference explained by a small variation in the number of parameters. On the other hand, there is a substantial reduction in MACs for the hybrid CDAE. This is one of the main advantages when a hybrid model is constructed directly from its real or complex counterpart. The layer sizes are reduced (fewer MACs) but compensated by additional layers (maintaining the same number of parameters). With this architecture, it is possible to efficiently outperform the baselines (real or complex) while still requiring fewer multiply-accumulate operations.

Table 7 MACs for each of the considered models. "R-" and "C-" indicate, respectively, MACs calculated from the real- and complex-valued branches of the neural network model

Model	Param.	R-MACs	C-MACs	MACs
rCDAE	173.3 k	4.72 G	0	4.72 G
cCDAE	171.5 k	0	4.54G	4.54 G
hCDAE	172.2 k	1.08 G	2.23G	3.31 G

As the hybrid CDAE allows us to use both real and complex outputs, we leverage this capability using prior knowledge from speech enhancement. Following similar considerations as the speech enhancement models presented in [28, 29], where the signal is denoised in two stages—a real-valued magnitude masking operation followed by a complex-valued filtering—we define the hCDAE architecture by using the real-valued branch to output a magnitude mask $\hat{m}_{mag}$. However, instead of finding filter coefficients with the complex-valued branch, we output a complex correction $\hat{s}_{cc}$ for the denoised speech $\hat{s}$. While the outputs of the rCDAE and cCDAE are applied as a pure complex mask, $\hat{s} = z \cdot \hat{m}$, the hCDAE output also provides the complex correction: $\hat{s} = z \cdot \hat{m}_{mag} + \hat{s}_{cc}$.

Note that, for this case, we do not follow a NAS procedure, as we aim to compare neural network models of the same size, which is outside the scope of the current experiment. However, for optimal performance, we recommend the NAS procedure from Section 4 and a detailed architecture based on the building block. Additionally, in this experiment, the HNN functional block described in Section 3 is not used per se, but as a macro-level version of the same concept. That can be seen in Fig. 12c when compared to Fig. 6, where the convolutional layers in the building block are replaced with convolutional encoders/decoders.

7.2.3 Performance

We trained the real-, complex- and hybrid-valued CDAE models for 100 epochs each, using the Adam optimizer with a weight decay of 0.0001 and an initial learning rate of 0.001, decaying exponentially to 0.0001 by the final epoch. Similar to [24], we use the (negative) SI-SDR as the minimisation loss, since it is a lower bound to both SDR and SNR [30] and has been shown to be a general and effective loss for monaural speech enhancement tasks [31].

We analyse the results in terms of SI-SDR, as well as the short-time objective intelligibility (STOI) measure [32] and the perceptual evaluation of speech quality (PESQ) score [33]. While PESQ evaluates speech quality and degrades significantly with distortion, the STOI metric is less affected by artefacts and focuses on the intelligibility of the speech signal.

The results are shown in Fig. 13. First, we confirm that all models successfully denoise the signal, improving each of the considered metrics compared to the noisy input. Furthermore, we observe that the real-valued network always underperforms its complex and hybrid counterparts. This aligns with observations from the classification experiment in Section 7.1, reflecting the inherent inefficiency of processing complex-valued data, such as learning to replicate a complex convolution as

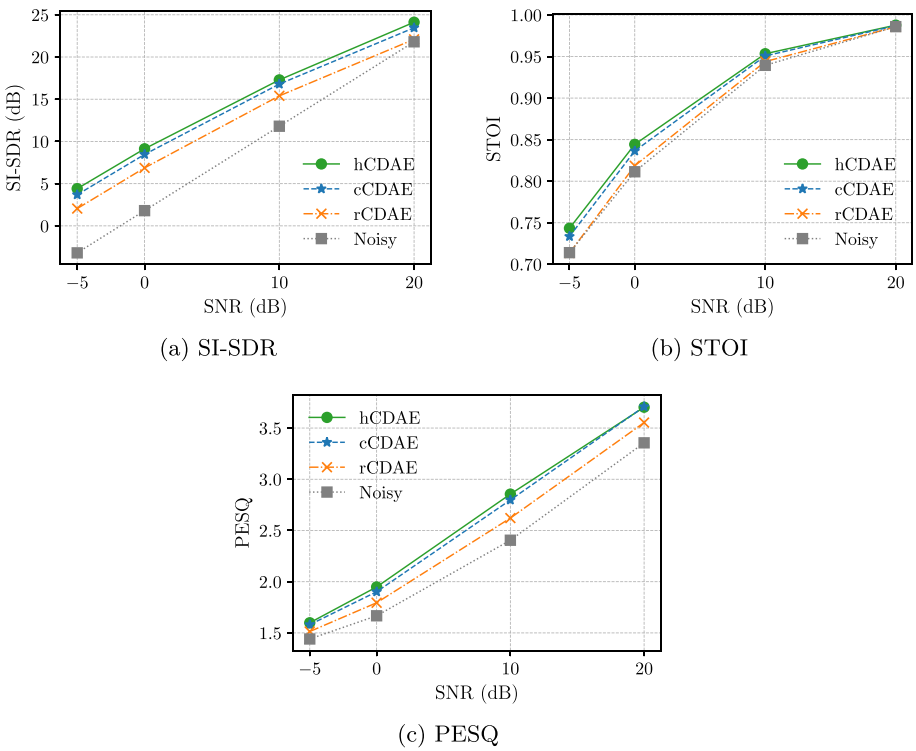


Fig. 13 SI-SDR, STOI and PESQ values obtained with the denoised speech signal by the rCDAE, cCDAE and hCDAE models over the test set for different SNR levels

demonstrated in the toy example of Section 2. On the other hand, the hybrid CDAE outperforms both rCDAE and cCDAE.

As hinted in prior works [1], the fully complex network applied to monaural speech enhancement has been shown to be quite inefficient, both in terms of training and complexity. Our results indeed indicate that the simple usage of complex layers, directly for estimating a mask and performing speech enhancement, is not an optimal solution. In the hCDAE model, nevertheless, the complex branch is used to produce correction coefficients to the final enhanced speech signal, which can directly compensate for the noisy phase. Additionally, the real branch performs the main filtering operation by estimating a magnitude mask. While this is based on prior knowledge and somewhat similar to state-of-the-art speech enhancement [28, 29], our approach also allows for both real and complex branches to share information via domain conversion. For this application, domain conversion maintains the synergistic operation of both branches: from the complex to the real domain, it informs magnitude processing about phase content; from real to complex domain, it enhances latent space features with magnitude filtering information.

Finally, the best performance is obtained with a substantially lower number of multiply-accumulate operations, validating the approach for practical applications in speech denoising. This is expected, given that the number of weights in convolutional layers is not linearly proportional to the number of MACs; more layers with smaller size result in lower complexity than fewer layers with larger size. This is consistent with the results observed for the classification task in Section 7.1 (Table 3).

8 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, an HNN could grow complex-valued pathways to expand the network and become more efficient in terms of loss and parameter storage, also presenting fewer multiply-accumulate operations, as shown in the speech denoising experiment. This represents a practical option for expanding existing real-valued networks to their hybrid counterpart. Moreover, the hybrid format introduces more layers with reduced size, lowering computational complexity in a manner analogous to mixture-of-experts in transformer architectures [34]. In practice, this translates to, for example, lower battery consumption for portable devices.

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.

The generalised activation functions (9a) and (9b) offer the expressivity of polynomials (when $|z|^q < \epsilon$) and self limiting from the denominator term (when $|z|^q > \epsilon$), this is an alternative to the use of splines. On the one hand, keeping the coefficients real-valued, we can generalise real-valued activation functions as bounded polynomials and on the other hand, using complex valued coefficients, generalisation into the complex space is facilitated. Inspired by Kolmogorov-Arnold network (KAN) architectures [35], HNNs can offer a middle ground between KAN, real- and complex-valued architectures by allocating coefficients between the linear (convolutions) and non-linear (activation) functions.

Expanding this work to other models like variational autoencoders [36], a potential approach is to modify the encoder and decoder into hybrid real- and complex-valued counterparts, using a block similar to Fig. 6. The latent space could include real-valued variances and real- and complex-valued means at the output of the encoder. In a similar way, for a transformer structure—initially proposed by [37]—processing blocks like convolutional and fully connected layers could be expanded into a hybrid version as in Fig. 6, 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 [38], 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 4, drastically reducing the number of parameters and improving the model's performance.

9 Conclusion

We have analysed how an RVNN processed complex-valued data and, with these insights, we have proposed an HNN architecture and a tailored neural architecture search procedure. Comparing an RVNN and an 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 native support for 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. The obtained results show great promise in terms

of increased performance and model size reduction with the hybrid architecture, which successfully leverages real- and complex-valued processing. In comparison to existing methods, the HNN can achieve state-of-the-art classification accuracy for the problem of spoken digits recognition with a dramatically lower number of parameters, due to the leveraging of complex operations allied to a dedicated NAS framework. Moreover, the proposed concept was also applied to speech enhancement showing that the hybrid model could achieve superior performance to its real- or complex-valued counterparts with a dramatic reduction in multiply-accumulate operations.

The hybrid architecture is proposed, either relying on an extensive and systematic neural architecture search procedure to reduce its size and optimise its performance or applied as a concept, which naturally results in best performance and lower complexity. As a future line of research in architecture development, a simplification of the model design can be investigated. Regarding the HNN application, 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.

Appendix 1: Complex-valued activation function analysis

In this appendix, we analyse each complex-valued activation function, starting with functions that have no intended real-valued provenance. We then demonstrate how complex-valued activation functions can be derived from real-valued counterparts. This Appendix supports Section 6.

To provide an intuitive visualisation of each complex-valued activation function, we define a reference function as $z = xe^{2i\pi x}$, where x varies continuously over the interval $\in [0, 3]$. This produces a spiral trajectory in the Cartesian plane, as shown in Fig. 14(A), also known as an Archimedean spiral, exhibiting a predictable magnitude of 0 to 3 and a phase variation spanning three rotations. The output z is used for the input of each complex-valued activation function. Linear operations will result in a similar spiral trajectory.

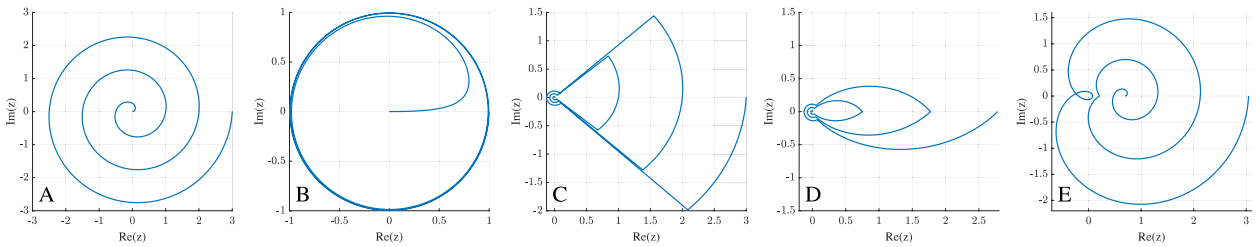


Fig. 14 Complex-valued activation functions. **A** reference function, **B** cRecip, **C** cReImLU, **D** cRecipMax, **E**: Cardioid

Function (B), defined as $Z_o = \frac{z}{|z|+0.01}$, is referred to as cRecip, representing the simplest reciprocal form used in this work. It originates from an early study by Georgiou and Koutsougeras [16] and is characterised by its near magnitude-agnostic behaviour.

Function (C), defined as $Z_o = z(0.05+0.95\text{sign}(\nu)), \nu = \max(0, \Re(z) - |\Im(z)| - 0.1)$ is referred to as cReImLU, introduces a deliberate phase-dependent magnitude switching effect as a complex-valued version of the ReLU function.

Function (D), defined as $Z_o = z(0.10+\frac{0.9\nu^2}{|z|^2+0.1}), \nu = \max(0, \Re(z) - 0.5|\Im(z)| - 0.1)$ is referred to as cRecipMax, includes a max function and serves as a smooth version of phase-dependent magnitude modulation.

Function (E), for comparison, is a Cardioid, mentioned in [7], we did not include this function in the NAS option list as it exhibits similar behaviour to functions Fig. 15(A and E).

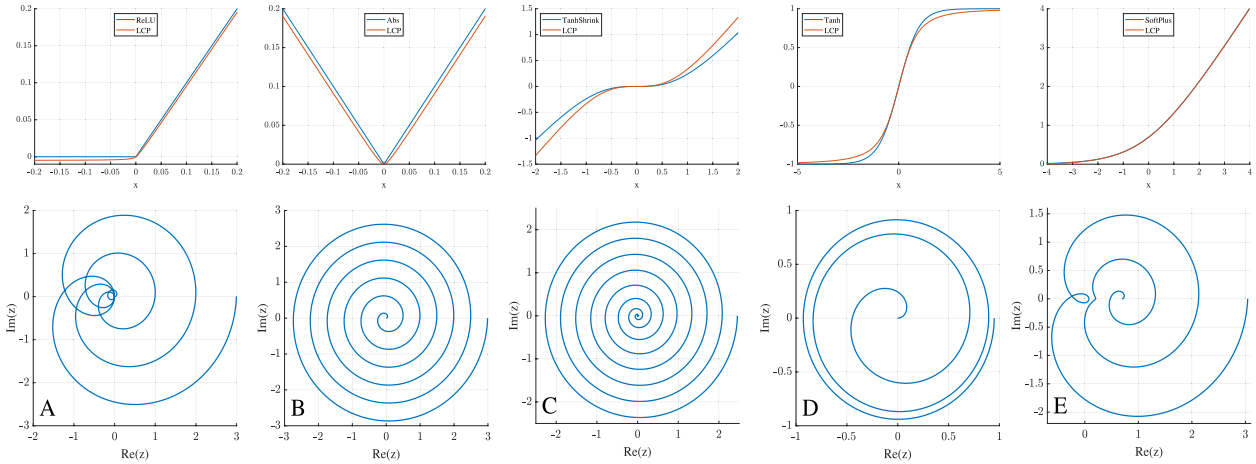


Fig. 15 Real-valued activation functions (above) and their derived complex-valued activation functions (below). **A** ReLU, **B** Abs, **C** TanhShrink, **D** Tanh, **E** Softplus

Moreover, we employ a class of functions referred to as *linear plus constrained polynomial* (LCP) for the remainder of the complex-valued activation function definitions, previously defined in Eqs. (9a) and (9b) and as a reminder here (12a) and (12b). In this formulation, the denominator serves to constrain the growth of the polynomial numerator when $|z|^q > \epsilon$, ensuring that the function approaches at most linear behaviour in the limit $z \to \infty$ with suitable values for p and q. Conversely, when $|z|^q < \epsilon$, the function's shape is dominated by the polynomial term and the linear component αz. ϵ must be greater than zero but not necessarily small.

$$Z_o = \alpha z + \frac{1}{|z|^q + \epsilon} \sum_{n=0}^{p} k_n z^n \tag{12a}$$

$$Z_o = \alpha z + \frac{1}{\sqrt{|z|^q + \epsilon}} \sum_{n=0}^{p} k_n z^n \tag{12b}$$

The ReLU activation function is extended to the complex domain by first expressing the real-valued function in terms of linear and absolute components, then reformulating it using linear and quadratic terms, as shown in Eq. (13a). Equation (13b) introduces a division protection term ϵ with its chosen value, resulting in the final LCP form. This, along with the original ReLU function is shown in Fig. 15(A, above). The transition from the real domain (x) to complex domain (z) is shown in Eq. (13c) and visualised in Fig. 15(A, below). A key consequence of using LCP functions is that their gradients are continuous, resulting in smooth functions. In the case of ReLU, the 'sharpness' of the transition around zero can be increased by reducing ϵ toward zero.

$$y = \text{ReLU}(x) = \frac{x}{2} + \frac{|x|}{2} = \frac{x}{2} + \frac{x^2}{2|x|} \tag{13a}$$

$$y = \frac{x}{2} + \frac{1}{2}\frac{x^2}{|x| + \epsilon}, \quad \epsilon = 0.01 \tag{13b}$$

$$Z_o = \frac{z}{2} + \frac{1}{2}\frac{z^2}{|z| + \epsilon} \tag{13c}$$

The Abs activation function is similarly extended to the complex domain by reformulating it using a quadratic term, as shown in Eq. (14a). Equation (14b) shows the inclusion of the division protection term ϵ with its chosen value and final complex-valued LCP form, using $z = x$ the real-valued input and output can be plotted with the original Abs function in Fig. 15(B, above). The complex domain plot is shown in Fig. 15(B, below).

$$y = |x| = \frac{x^2}{|x|} \tag{14a}$$

$$Z_o = \frac{z^2}{|z| + \epsilon}, \quad \epsilon = 0.01 \tag{14b}$$

The TanhShrink activation function can be extended to the complex domain by reformulating the real-valued function using either an approximation of tanh (see Eq. (15a)) or a cubic expression (see Eq. (15b)). Since the tanh term will be used later, we opted for the cubic form

here to diversify the function set. Equation (15c) shows the final complex-valued LCP form and sets the value of ϵ. By setting $z = x$ the real-valued input and output can be plotted with the original TanhShrink function in Fig. 15(C, above). The complex domain plot is shown in Fig. 15(C, below).

$$y = x - tanh(x) \approx x - \frac{x}{|x| + \epsilon} \tag{15a}$$

$$y = x - tanh(x) \approx \frac{x^3}{|x|^2 + \epsilon} \tag{15b}$$

$$Z_o = \frac{z^3}{|z|^2 + \epsilon}, \quad \epsilon = 2 \tag{15c}$$

The Tanh activation function can be extended to the complex domain by reformulating the real-valued approximation of tanh (see Eq. (16a)). Equation (16b) shows the final complex-valued LCP form and sets the value of ϵ. By setting $z = x$ the real-valued input and output can be plotted with the original Tanh function in Fig. 15(D, above). The complex domain plot is shown in Fig. 15(D, below).

$$y = tanh(x) \approx \frac{x}{\sqrt{|x|^2 + \epsilon}} \tag{16a}$$

$$Z_o = \frac{z}{\sqrt{|z|^2 + \epsilon}}, \quad \epsilon = 1 \tag{16b}$$

The Softplus activation function can be extended to the complex domain by fitting to the real-valued LCP function, Eq. (17a). To ensure overall shape support, the following conditions are applied: $\lim_{x\to\infty} \frac{dy}{dx} = 1$ and $\lim_{x\to-\infty} \frac{dy}{dx} = 0$. This already determines that $\alpha = k_2 = 0.5$ and the remaining variables k_0 and ϵ are easy to fit using numerical methods. Equation (17b) shows the final complex-valued LCP form with all variables set. By setting $z = x$ the real-valued input and output can be plotted with the original Tanh function in Fig. 15(E, above). The complex domain plot is shown in Fig. 15(E, below).

$$y = \log(1 + \exp(x)) \approx \alpha x + \frac{k_0 + k_2 x^2}{\sqrt{|x|^2 + \epsilon}} \tag{17a}$$

$$Z_o = 0.5z + \frac{2.134 + 0.5z^2}{\sqrt{|z|^2 + 9.481}} \tag{17b}$$

In selecting the eight complex-valued activation functions for NAS, we aimed to maximise diversity in functional scope. The real-valued activation function derivations incorporate angular, smooth, symmetrical and asymmetrical forms, and both types of LCP functions, as shown in Eqs. (12a) and (12b), the latter including a square root in its denominator. The non-derived functions (see Fig. 14(B, C and D)) exploit angular and smooth phase modulation of magnitude, as well as magnitude-dependent modulation of magnitude. One may ask: why are these phase-dependent functions orientated in the same direction, rather than at different phase angles? Since geometric translation, scaling and rotation are linear operations, complex-valued convolution layers inherently account for such transformations.

We have deliberately chosen not to include *split* complex-valued activation functions, where the real and imaginary parts are processed separately. Despite their simplicity of implementation, split activations have notable disadvantages: they disregard magnitude and phase relationships, limiting expressiveness, and may suffer from gradient mismatch, leading to suboptimal learning. If a model needs to process these components independently, they should be converted back to the real domain and treated as separate signals. This is where hybrid networks offer a clear advantage over purely real-valued networks.

Authors' contributions
Alex Young conceptualised the study, designed the methodology, interpreted results, wrote original software, conducted experiments and wrote the original draft. Luan Vinícius Fiorio designed the methodology, wrote original software, conducted experiments and wrote the original draft. Bo Yang performed software development, execution of experiments, analysis of results and wrote the original draft. Boris Karanov provided critical feedback and revision. Wim van Houtum provided supervision, critical feedback and project administration. Ronald M. Aarts provided supervision and critical feedback.

Funding
This work was supported by the Robust AI for SafE (radar) signal processing (RAISE) collaboration framework between Eindhoven University of Technology and NXP Semiconductors, including a Privaat-Publieke Samenwerkingentoeslag (PPS) supplement from the Dutch Ministry of Economic Affairs and Climate Policy.

Data availability
The data necessary for reproduction of results is publicly available at https://github.com/soerenab/AudioMNIST [23]. The proposed methods and experiments can be completely reproduced based on the text, equations, figures and tables in the manuscript. The tools used to obtain results, Python programming language (https://www.python.org) and PyTorch (https://pytorch.org) [39], are free to use and publicly available.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 14 March 2025 Accepted: 5 March 2026
Published online: 09 April 2026

References

1. H. Wu, K. Tan, B. Xu, A. Kumar, D. Wong. Rethinking complex-valued deep neural networks for monaural speech enhancement, [v1], (2023). https://doi.org/10.48550/arXiv.2301.04320
2. T. Peer, T. Gerkmann, Phase-aware deep speech enhancement: it's all about the frame length. JASA Express Lett. **2**(10), 104802 (2022). https://doi.org/10.1121/10.0014875
3. A.M. Sarroff, Complex Neural Networks for Audio – digitalcommons.dartmouth.edu. Phd thesis, Dartmouth College (2018)
4. C. Lee, H. Hasegawa, S. Gao, Complex-valued neural networks: a comprehensive survey. IEEE/CAA J. Autom. Sin. **9**(8), 1406–1426 (2022)
5. A. Fuchs, J. Rock, M. Toth, P. Meissner, F. Pernkopf, in *2021 IEEE Radar Conference (RadarConf21)*. Complex-valued convolutional neural networks for enhanced radar signal denoising and interference mitigation (2021), pp. 1–6. https://doi.org/10.1109/RadarConf2147009.2021.9455296
6. S. Welker, J. Richter, T. Gerkmann, in *Proc. Interspeech 2022*. Speech Enhancement with Score-Based Generative Models in the Complex STFT Domain, Interspeech. (2022), pp. 2928–2932. https://doi.org/10.21437/Interspeech.2022-10653
7. P. Virtue, S.X. Yu, M. Lustig, in *2017 IEEE International Conference on Image Processing (ICIP)*. Better than real: Complex-valued neural nets for mri fingerprinting (2017), pp. 3953–3957. https://doi.org/10.1109/ICIP.2017.8297024
8. Y. Quan, Y. Chen, Y. Shao, H. Teng, Y. Xu, H. Ji, Image denoising using complex-valued deep cnn. Pattern Recognition **111**, 107639 (2021)
9. C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J.F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, C.J. Pal, in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. Deep complex networks (2018). https://doi.org/10.48550/arXiv.1705.09792
10. C.A. Popa, in *2018 International Joint Conference on Neural Networks (IJCNN)*. Deep hybrid real-complex-valued convolutional neural networks for image classification (2018), pp. 1–6. https://doi.org/10.1109/IJCNN.2018.8489274
11. H. Du, R.P. Riddell, X. Wang, A hybrid complex-valued neural network framework with applications to electroencephalogram (EEG). Biomed. Signal Process. Control **85**, 104862 (2023)
12. Y. Ju, J. Chen, S. Zhang, S. He, W. Rao, W. Zhu, Y. Wang, T. Yu, S. Shang, in *ICASSP 2023–2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Tea-pse 3.0: Tencent-ethereal-audio-lab personalized speech enhancement system for icassp 2023 dns-challenge (2023), pp. 1–2
13. S. Lv, Y. Hu, S. Zhang, L. Xie. Dccrn+: Channel-wise subband dccrn with snr estimation for speech enhancement, [v1], (2021). https://doi.org/10.48550/arXiv.2106.08672
14. T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Optuna: A next-generation hyperparameter optimization framework, KDD '19 (Association for Computing Machinery, New York, 2019), pp. 2623–2631
15. J. Bassey, L. Qian, X. Li, A survey of complex-valued neural networks (2021)
16. G. Georgiou, C. Koutsougeras, Complex domain backpropagation. IEEE Trans. Circuits Syst. II Analog Digit. Signal Process. **39**(5), 330–334 (1992)
17. M. Sipper, Neural networks with À la carte selection of activation functions. SN Comput. Sci. (2021), p. 3. https://doi.org/10.1007/s42979-021-00885-1
18. P. Bacher. Audio mnist digit vocal recognition. Kaggle Notebook (2023). https://www.kaggle.com/code/paulbacher/audio-mnist-digit-vocal-recognition. Accessed 12 Mar 2025
19. V. Tsouvalas, A. Saeed, T. Ozcelebi, in *ICASSP 2022–2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Federated self-training for data-efficient audio recognition (2022), pp. 476–480. https://doi.org/10.1109/ICASSP43922.2022.9746356
20. D. Hayakawa, T. Masuko, H. Fujimura, in *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. Applying complex-valued neural networks to acoustic modeling for speech recognition (IEEE, 2018), pp. 1725–1731. https://doi.org/10.23919/APSIPA.2018.8659610
21. A. Thite. Sound mnist: Spoken digit classification with cnns. GitHub Repository (2023). https://github.com/adhishthite/sound-mnist. Accessed 12 Mar 2025
22. C. Gawande. Audio classification with the audio mnist dataset. Medium (2023). https://medium.com/@cgawande12/audio-classification-with-the-audio-mnist-dataset-0ad95c3fb713. Accessed 12 Mar 2025
23. S. Becker, J. Vielhaben, M. Ackermann, K.R. Müller, S. Lapuschkin, W. Samek, AudioMNIST: exploring explainable artificial intelligence for audio analysis on a simple benchmark. J. Frankl. Inst. **361**(1), 418–428 (2024). https://doi.org/10.1016/j.jfranklin.2023.11.038
24. L.V. Fiorio, B. Karanov, B. Defraene, J. David, F. Widdershoven, W. Van Houtum, R.M. Aarts, Spectral masking with explicit time-context windowing for neural network-based monaural speech enhancement. IEEE Access **12**, 154843–154852 (2024). https://doi.org/10.1109/ACCESS.2024.3483443
25. H. Zen, V. Dang, R. Clark, Y. Zhang, R.J. Weiss, Y. Jia, Z. Chen, Y. Wu, LibriTTS: A corpus derived from librispeech for text-to-speech. arXiv preprint 1904.02882 (2019). https://arxiv.org/abs/1904.02882. [cs.SD]. Accessed July 2025.
26. A. Mesaros, T. Heittola, T. Virtanen, in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*. A multi-device dataset for urban acoustic scene classification (2018), pp. 9–13
27. D.S. Williamson, Y. Wang, D. Wang, in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Complex ratio masking for joint enhancement of magnitude and phase (2016), pp. 5220–5224. https://doi.org/10.1109/ICASSP.2016.7472673
28. H. Schroter, A.N. Escalante-B, T. Rosenkranz, A. Maier, in *ICASSP 2022–2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Deepfilternet: A low complexity speech enhancement framework for full-band audio based on deep filtering (2022), pp. 7407–7411. https://doi.org/10.1109/ICASSP43922.2022.9747055
29. H. Schröter, A. Maier, A. Escalante-B, T. Rosenkranz, in *2022 International Workshop on Acoustic Signal Enhancement (IWAENC)*. Deepfilternet2: Towards real-time speech enhancement on embedded devices for full-band audio (2022), pp. 1–5. https://doi.org/10.1109/IWAENC53105.2022.9914782
30. J.L. Roux, S. Wisdom, H. Erdogan, J.R. Hershey, SDR - half-baked or well done? arXiv preprint [v1]. (2018) [cs.SD]. https://doi.org/10.48550/arXiv.1811.02508
31. M. Kolbæk, Z.H. Tan, S.H. Jensen, J. Jensen, On loss functions for supervised monaural time-domain speech enhancement. IEEE/ACM Trans. Audio Speech Lang. Process. **28**, 825–838 (2020). https://doi.org/10.1109/TASLP.2020.2968738
32. C.H. Taal, R.C. Hendriks, R. Heusdens, J. Jensen, in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. A short-time objective intelligibility measure for time-frequency weighted noisy speech (2010), pp. 4214–4217. https://doi.org/10.1109/ICASSP.2010.5495701
33. A. Rix, J. Beerends, M. Hollier, A. Hekstra, in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*. Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs, vol. 2 (2001), pp. 749–75. https://doi.org/10.1109/ICASSP.2001.941023
34. S. Mu, S. Lin, A comprehensive survey of mixture-of-experts: Algorithms, theory, and applications [v3]. (2025). https://arxiv.org/abs/2503.07137
35. Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T.Y. Hou, M. Tegmark. Kan: Kolmogorov-arnold networks [v4]. (2024). https://doi.org/10.48550/arXiv.2404.19756
36. L. Pinheiro Cinelli, M. Araújo Marins, E.A. Barros da Silva, S. Lima Netto, *Variational Autoencoder* (Springer International Publishing, Cham, 2021), pp. 111–149
37. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez et al., in *Advances in neural information processing systems*. Attention is all you need (Google, 2017), 31st Conference on Neural Information Processing Systems, https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
38. V. Tokala, E. Grinstein, M. Brookes, S. Doclo, J. Jensen, P.A. Naylor, in *ICASSP 2024–2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Binaural speech enhancement using deep complex

convolutional transformer networks (2024), pp. 681–685. https://doi.org/10.1109/ICASSP48485.2024.10447090

39. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, *Pytorch: An imperative style, high-performance deep learning library, [v1]* (2019). https://arxiv.org/abs/1912.01703

Publisher's Note