TWO INSTANCES OF INTERPRETABLE NEURAL NETWORK FOR UNIVERSAL APPROXIMATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper proposes two bottom-up interpretable neural network (NN) constructions for universal approximation, namely Triangularly-constructed NN (TNN) and Semi-Quantized Activation NN (SQANN). The notable properties are (1) resistance to catastrophic forgetting (2) existence of proof for arbitrarily high accuracies on training dataset (3) for an input x, users can identify specific samples of training data whose activation "fingerprints" are similar to that of x's activations. Users can also identify samples that are out of distribution.

1 INTRODUCTION

Artificial neural networks (NN) have recently seen successful applications in many fields. Modern deep neural network (DNN) architecture, usually trained through the backpropagation mechanism, has been called a black-box because of its lack of interpretability. To tackle this issue, various studies have been performed to understand how a NN works; see the following surveys Arrieta et al. (2020); Gilpin et al. (2018); Tjoa & Guan (2020); Wiegreffe & Marasović (2021). This paper primarily proposes two interpretable models to perform universal approximation with three main desirable properties, complete with mathematical proofs and experimental assessments on some regression and classification problems.

Related works and interpretability issues. Recent remarkable studies on universal approximators include the detailed results on deep narrow network by Kidger & Lyons (2020), DeepONet universal approximation for operators by Lu et al. (2021) and the universal approximation of the Broad Learning System by Chen et al. (2019); also see other relevant works with remarkable results e.g. Hanin (2019); Park et al. (2021); Johnson (2019). While insightful, they do not directly address the eXplainable Artificial Intelligence (XAI) issue, especially the blackbox property of the DNN. Similarly, a number of classical papers provide theoretical insights for NN as universal approximators, but interpretability, transparency and fairness issues were not yet in the spotlight. The universal approximation theorem by Cybenko (1989) asserts that a NN with a single hidden layer can approximate any function to arbitrarily small error under common conditions. The proof makes use of classical mathematical theorems in the function space, where density of the space is often the concern. In particular, its theorem 1 uses an abstract proof by contradiction. From the proof, it is not easy to observe the internal mechanism of a NN in a straight-forward manner. Bottom-up constructions for function approximation using NN then emerged. They still do not focus on interpretability, thus readers have to observe for themselves the shape of the resulting networks from the components used in the construction. For example, there are works related to spline functions like Mhaskar & Micchelli (1992); Mhaskar (1993b;a); Chui et al. (1994); another example by Sartori & Antsaklis (1991) where two-layer NN can be formed by arbitrarily choosing the weights of first layer and computing the weights of second layer explicitly; and section 5 of Pinkus (1999) shows the equations used to obtain weights and proves the existence of such solutions. In some of them and others such as Mhaskar (1996); Chui et al. (1996), the focus lies in error quantification.

Outline. This paper provides the constructions of (1) triangularly-constructed NN (TNN), interpretable through linear ordering of activations (2) Semi-Quantized Activation NN (SQANN) whose neurons in each layer can be understood as the exact fingerprint of a training data sample, enhancing its interpretability. TNN will be the simpler construction that inspires SQANN. We must warn readers that our constructions are rooted in conditional algorithms, thus they might be somewhat technical; we provide pedagogical examples to help clarify the details. Their special features are:

1. Resistance to *catastrophic forgetting*. See below.
2. Both have mathematical proofs for arbitrarily high accuracy on training datasets, unlike popular DNN which are often empirical.
3. for each prediction, strongly activated neurons or half-activated neurons (if any) can be identified precisely (see fig. 1(C) bottom for SQANN and fig. 2 for TNN). SQANN also allows user to identify a sample that activates all nodes weakly, i.e. the sample is likely out of distribution.

Regarding resistance to *catastrophic forgetting*. Unlike artificial NN, mammalian brain retains old information when it learns new information *by protecting previously acquired knowledge in neocortical circuits*; see Kirkpatrick et al. (2017) and the references thereof. As rats learn new skills, the volumes of dendritic spines in their brains increase (Yang et al. (2009)) while existing dendrites persist, thus they retain old memories. Both TNN and SQANN have similar property. Particularly, SQANN increases the size of a layer as it progressively acquires new samples during construction (learning). This inevitably increases the number of weights that connect the layers; see fig. 1(D.1, D.2). Furthermore, each old sample is not forgotten since its exact "fingerprint" is already registered (i.e. input x has been converted) to a neuron's nucleus $\eta_l^{<k>}$, shown in fig. 1(B). The activation pattern of a sample x in $SQANN(x)$ includes the value 1 in a specific node of a "synapse", $v_l^{<k>}$ and the particular combinations of values in other synapses.

This paper is arranged as the following. Section 2 shows TNN construction, related results, including *a pencil-and-paper example* for pedagogical purpose. Section 3 shows SQANN construction, statements regarding SQANN, *another pencil-and-paper example*, then experimental results of its application, before we conclude the paper with limitations and future works. Python codes and clearer images of figures are fully available in supp. materials (also see appendix).

2 TRIANGULARLY-CONSTRUCTED NN (TNN)

TNN is the prototype NN for our interpretable universal approximator that we will extend to the more general construct, SQANN. It provides a manageable illustration for (1) organized activations of neurons and (2) the retrieval of α values as the outputs. TNN is in the form $TNN(x) = \alpha^T \sigma(Wx + b)$ where $x \in [0, 1]^n$, $\alpha, b \in \mathbb{R}^N$ and $W \in \mathbb{R}^{N\times n}$, where we use sigmoid function σ for simplicity. It is constructed on a linearly ordered dataset containing N samples $\{(x^{(k)}, y^{(k)}) \in \mathbb{R}^n \times \mathbb{R}\}_{k=1}^N$ such that $x^{(N)} < x^{(N-1)} < \cdots < x^{(1)}$ and $y^{(k)} = f(x^{(k)})$, f the true function that TNN will approximate. We start with a simple scalar function $y = f(x) \in \mathbb{R}$ for $x \in [0, 1]$, thus TNN's interpretability can be illustrated very clearly. The interpretability comes from the linear ordering property where higher value of x (≈ 1) will activate more neurons while lower values will activate less neurons as shown in fig 2(A). Then α values will be retrieved in a continuous way through dot product, eventually used to compute the output for prediction. For more remarks and *linear ordering* concept in mathematics, see appendix.

Ordered activation. We would like $x^{(1)}$ to activate all neurons, while $x^{(N)}$ activates only 1 neuron. In other words, ideally $\sigma(Wx^{(1)} + b) = [1, 1, \ldots, 1, 1]^T$, followed by $\sigma(Wx^{(2)} + b) = [1, 1, \ldots, 1, 0]^T$ and so on until $\sigma(Wx^{(N)} + b) = [1, 0, \ldots, 0]^T$; again, refer to fig. 2(A). With this concept, we seek to achieve interpretability by successive activations of neurons depending on the "intensity" of the input, with $x^{(1)}$ being the most intense. In general, the above can be written as

$$\sigma^{(k)} \equiv \sigma(Wx^{(k)} + b) = [\underbrace{1, \ldots, 1}_{N-(k-1)}, \underbrace{0, \ldots, 0}_{k-1}]^T \tag{1}$$

which is approximately achieved for $k = 1, \ldots, N$ at large a (and exactly if $a \to \infty$) with

$$(Wx^{(k)} + b)_j = \begin{cases} \leq -a, j \geq N - k + 2 \\ \geq +a\ , j \leq N - k + 1 \end{cases} \tag{2}$$

TNN construction: computing weights W, b, α. How then do we compute $W,\ b$ to achieve the *ordered activation*? Consider first $(Wx^{(2)}+b)_N = -a$ and $(Wx^{(1)}+b)_N = a$ and solve them. This yields $W_N = 2a/\Delta^{(1)}$ and $b_N = a - W_N x^{(1)}$ where $\Delta^{(1)} = x^{(1)} - x^{(2)}$. Iterating through k, i.e. solving $(Wx^{(k+1)} + b)_{N-k+1} = -a$ and $(Wx^{(k)} + b)_{N-k+1} = a$ we obtain $W_{N-k+1} = 2a/\Delta^{(k)}$ and $b_{N-k+1} = a - W_{N-k+1}x^{(k)}$ where $\Delta^{(k)} = x^{(k)} - x^{(k+1)}$.

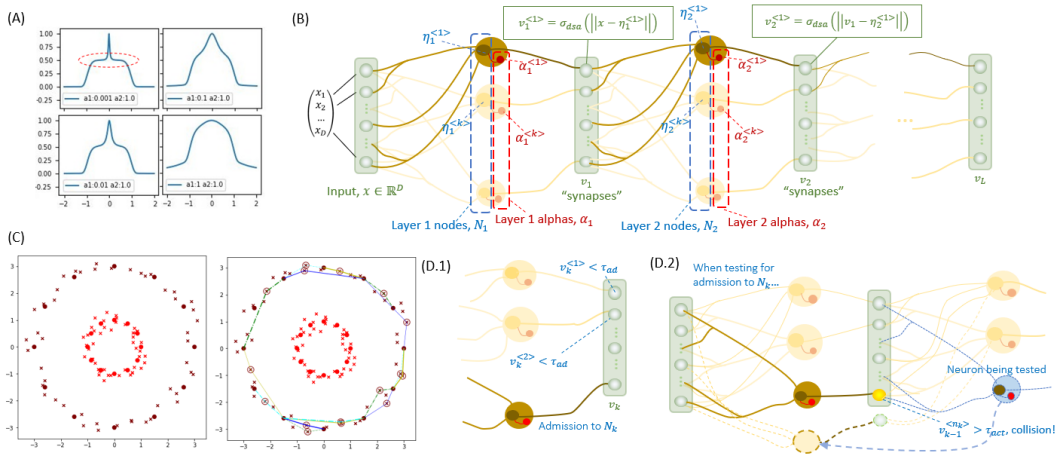

Figure 1: (A) Double selective activation with different parameters. (B) SQANN schematic. Each layer (N_k, α_k) is stylized as a collection of neurons. A neuron stores the main "fingerprint" in nucleus $\eta_l^{<k>}$ (dark brown) and its corresponding "output" in nucleus $\alpha_l^{<k>}$ (dark red). When strong activation is detected, the signal will be redirected to the dark red nucleus $\alpha_l^{<k>}$. (C) SQANN used for a simple classification. (Left) The large filled dots are training samples, x marks are test samples. Bright red indicates $y = 1.0$, dark red $y = 0.5$. (Right) Same as left but test samples that are interpolated (i.e. no strong activation) are annotated with red open circles; colored lines indicate which two training samples are used for the interpolation. Lines are marked with different colors and styles for clarity. (D) Construction of SQANN when (D.1) admission occurs: a new neuron is introduced, creating more connection analogous to mammalian brains. (D.2) collision occurs.

For $\Delta^{(N)}$, we need a dummy $x^{(N+1)}$ value or we can directly choose its value, for example, by using the average value $\frac{1}{N}\Sigma_{k=1}^{N-1}\Delta^{(k)}$. The effect is illustrated by the value near $x = 0$ in fig. 3(A1-3) and should not pose any problem; the chosen dummy value will only affect the shape at the left end of the graph. We can rewrite the indices so that $W_k = 2a/\Delta^{(N-k+1)}$ and $b_k = a - W_k x^{(N-k+1)}$ whenever convenient.

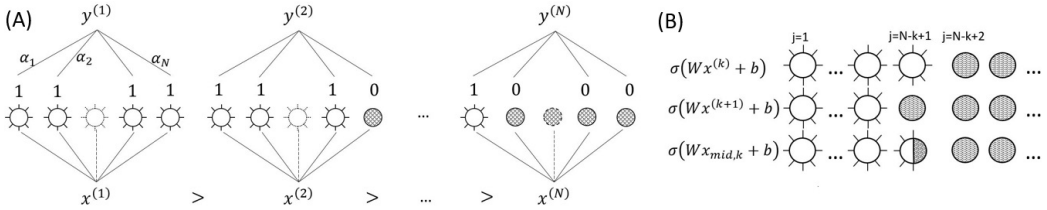

Figure 2: (A) Triangular construction is built by prioritizing interpretability of a neural network. As $x^{(k)}$ decreases in "strength", the neurons are "turned off" correspondingly. (B) Activations of neurons for $x^{(k)}, x^{(k+1)}$ and their mid-point $x_{mid,k}$. Not only will neuron activation be half at the mid-point, the output $y_{mid,k} = \frac{1}{2}(y^{(k)} + y^{(k+1)})$ is also half the sum of its neighbours'.

We compute α using the property of equation (1). From fig. 2(A), this means ideally $y^{(1)} = \Sigma_{i=1}^{N}\alpha_i\sigma(Wx^{(1)} + b)_i$ for $a \to \infty$, and similarly $y^{(2)} = \Sigma_{i=1}^{N-1}\alpha_i\sigma(Wx^{(2)} + b)_i$ and so on until $y^{(N)} = \alpha_1\sigma(Wx^{(N)} + b)_1$. Putting them together as $y = [y^{(1)}, \ldots, y^{(N)}]^T$, we get $y = A\alpha$ where A is an upper-left triangular matrix and the inverse A^{-1} exists. Thus, $\alpha = A^{-1}y$, a matrix such that $A_{ij}^{-1} = 1$ along the diagonal, $A_{i,i+1}^{-1} = -1$ and zeroes otherwise, which facilitates a convenient computation. The triangular construction is complete:

$$TNN(x) = \alpha^T\sigma(Wx + b) \tag{3}$$

When the dataset is evenly spaced, $x^{(k)} = 1 - (k-1)\Delta, k = 1, ..., N, \Delta = 1/(N-1)$, the results simplify to $W_k = 2a(N-1)$ and $b_k = a(3-2k)$. Not only equation (2) is fulfilled, we also get $(Wx^{(k)} + b)_j = a(1 + 2[N - k + 1 - j])$. For the k-th data sample, the activation will then be well-spaced in an interval of $2a$, so that $\sigma^{(k)} = \sigma([\dots, -3a, -a, a, 3a, \dots])^T \approx [\dots, 0, 0, 1, 1, \dots]^T$.

TNN pencil-and-paper example. Use TNN to fit the dataset $(x, y) \in \{(1, 1), (0.5, 2), (0, 3)\}$. Then $f(x) \approx TNN(x) = 3\sigma(20x + 5) - \sigma(20x - 5) - \sigma(20x - 15)$. Full workings, see appendix.

Theorem 1 *TNN achieves arbitrarily high accuracy on the training dataset. Proof: see appendix.*

Smoothness. From the construction, assuming sigmoid function as the activation function, it is obvious that the function is continuous for finite a. As a increases, the function becomes more and more constant around each data sample as shown in fig. 3(A1-3), i.e. becoming more step-wise.

Generalizability to test dataset. The proof of theorem 1 shows perfect fitting to training dataset. There is an upper bound to errors on test dataset if the test dataset is well-behaved. Otherwise, if test dataset contains errant or novel samples, they can be incorporated into the training dataset to create a better model with the above-said error upper-bound, as shown in the following proposition:

Proposition 1 ***Errors on monotonous interval***. *Given finite training, test datasets D, D', there exists $A \subseteq D'$ such that, using TNN constructed with $D \cup A$, for all samples in test dataset $(x', y') \in D'$, sample-wise error $e = |y' - TNN(x')|$ has an upper bound $max(|y' - y^{(k+1)}|, |y' - y^{(k)}|)$ for some k. Proof: see appendix.*

Furthermore, there is a mid-point property that can be exploited for generalizability to arbitrarily high accuracy, where data must be sampled such that any instance x_{test} lies inside either (1) the training dataset or (2) is equal to some mid-point of two neighbouring training samples; see the proposition below. Fig. 2(B) shows how the component of $x_{mid,k}$ at $j = N - k + 1$ is half-activated i.e. the activation value is 0.5. Admittedly, this is an ideal condition for accurate generalizability.

Proposition 2 ***Mid-point property***. *The mid-point $x_{mid,k} = \frac{1}{2}(x^{(k)} + x^{(k+1)})$ takes the value of $\alpha^T \sigma(Wx_{mid,k} + b) = \frac{1}{2}(y^{(k)} + y^{(k+1)})$. Proof: see appendix.*

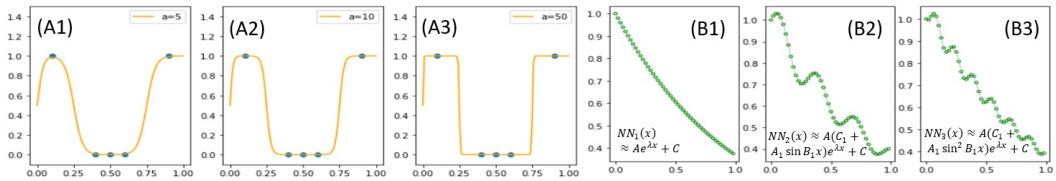

Figure 3: (A1-3) Three triangular constructions (orange plots) using different values of a. Higher a results in more step-wise plots and more constant values around the data samples (blue points). (B1) Plots of NNs approximated using triangular construction (smooth green plots) over scatter plots of the corresponding true data (green open circles). The parameters are as the following $A = 1, \lambda = -1, C = 0$ for all, (B2) $A_1 = 0.1, B_1 = 20, C_1 = 1$ (B3) $A_1 = 0.1, B_1 = 10, C_1 = 1$.

Generalization to n-dimensions. Generalization to scalar input and multi-dimensional output is relatively simple. From equation (3), we can treat α as the coefficients for the only component of one-dimensional y. Generalizing to $y \in \mathbb{R}^m, m > 1$, identify each vector α_i with the component y_i. Stacking them up, we can redefine $\alpha = [\alpha_1^T; \alpha_2^T; \dots]$ where a semi-colon denotes the next row, and the construction is done. Note that now $\alpha \in \mathbb{R}^{m \times N}$. Our further attempts at generalization to n-dimensional input data have been unsatisfactory, but do consider such attempts in the appendix. Still, along the way we ask, for high dimensional dataset, does there exist any linear ordering so that TNN can be used for high accuracy classification? The proposition indicates that there is.

Proposition 3 *Given a standard DNN for C classes classification with a_{tr} training accuracy, then there exists a linear ordering for TNN to achieve a_{tr} accuracy. Test accuracy a_{test} of DNN can be achieved by TNN with high probability through squeezed linear ordering. Proof: see appendix.*

3 SEMI-QUANTIZED ACTIVATION NEURAL NETWORK (SQANN)

Extending TNN to a multi-dimensional model directly turns out to be subtly difficult. After several trial-and-errors, we come up with the SQANN architecture which (1) retains TNN's idea of using sequence of activations to retrieve α, (2) remotely resembles a Radial Basis Function, but (3) still has some properties of deep neural network, such as the possibility of deep learning (multiple constructed layers) and neuron activations. The difference is, a neuron in SQANN corresponds exactly to a data sample as SQANN stores its "fingerprints" as neurons' nuclei -see fig. 1(B)- with different kinds of responses: (1) distinct peaks and (2) half-activations and (3) weak/zero activations, made possible by double selective activation σ_{dsa}, see fig. 1(A). This gives each SQANN neuron interpretability, at least in the following sense: samples are highly/moderately/not recognizable if their activation patterns strongly/moderately/weakly resemble the activation patterns of an existing training sample x, where identifications are facilitated by the distinct regions of σ_{dsa}. The design is "semi" quantized since σ_{dsa} has approximately "distinct" levels yet remains continuous. Furthermore, SQANN incorporates multi-layer structure, thus avoids non-generalizing structure that nearest neighbours methods suffer from; see the remark in (Pedregosa et al. (2011a)).

Notations. The order of data sample within the dataset matters, thus we define our own indexing to prevent confusion. Let the finite training set be $\{(x^{(k)}, y^{(k)}) \in X \times Y : k = 1, 2, \ldots, N\}$. We create the SQANN model that predicts $y^{(k)} = SQANN(x^{(k)})$ with provably perfect accuracy and generalizes well to similar test distribution. Subscript indicates layer, v denotes activation values collected in the "synapses", square bracket with subscript denotes vector component so that $[v_2]_4$ is the 4-th vector component of the activation of layer 2. **Layer k** consists of (N_k, α_k), where $N_k = (\eta_k^{<1>}, \eta_k^{<2>}, \ldots, \eta_k^{<n_k>})$ stores fingerprints/patterns, $\alpha_k = (y_k^{<1>}, y_k^{<2>}, \ldots, y_k^{<n_k>})$ stores output values. The angle bracket denotes the index after relabelling. Hence, if k-th data sample before relabelling is $(x^{(k)}, y^{(k)})$, $k = 50$ and it becomes the first node in layer 2, then we write $\eta_2^{<1>} = \eta_2^{(50)}$. **Concatenation**. To denote the addition of the new k-th node to the layer l, use $\eta_l^{<k>} \leftarrow v$, where v can be for example $v_2^{(m)}$ the activation of the m-th data at layer 2. Alternatively, $N_l \rightarrow concat(N_l, \eta_l^{<k>})$. We can speak about layer k using N_k if α_k is not yet involved. However, once concatenation of N_l is decided, always correspondingly concatenate the α_l i.e. $y_l^{<k>} \leftarrow y^{(m)}$. **Selective clustering** of $p_k = (x_k, y_k)$ for $k = 1, 2$ is loosely defined for x_1, x_2 that are close to each other such that: if y_1, y_2 are similar, then p_1, p_2 are clustered together, otherwise two distinct clusters are created; see appendix for formal definition, its effects on interpolation and more remarks.

Double selective activation. Given selective activation $\pi(x, a) = \frac{a}{a+x^2}$ and Super Gaussian $s_g(x, a) = exp(-(x/a)^{2n})$, $n = 4$,$r = 0.5$, then the *double selective activation* is (fig. 1(A)):

$$\sigma_{dsa}(x, a_1, a_2, r) = (1 - r) \times \pi(x, a_1) + r \times s_g(x, a_2) \tag{4}$$

Nodes activation. Denote the "synapse" or the activation value of node j at layer k by input v as:

$$[v_k]_j = \sigma_{dsa}\big(||v - \eta_k^{<j>}||\big) \tag{5}$$

where $\eta_k^{<j>} \in N_k$ for $j = 1, \ldots, n_k$ and n_k is the number of neurons/nodes in the layer. See fig. 1(B). In SQANN, activations will be forwarded layer by layer, i.e. $[v_1]_j = \sigma_{dsa}\big(||x - \eta_1^{<j>}||\big)$ where $\eta_1^{<j>} \in N_1$ and $[v_{k+1}]_j = \sigma_{dsa}\big(||v_k - \eta_{k+1}^{<j>}||\big)$ where $v_{k+1}^{<j>} \in N_{k+1}$.

3.1 SQANN CONSTRUCTION

SQANN is constructed without optimization like gradient descent. Each indexed training data sample is converted into a "fingerprint" or pattern of neuron activations. Outline of SQANN construction with interpretations (illustrated in the sketch of proof for proposition 4, appendix):

1. *Admission to layer k*. Sample's new/distinct fingerprint is added into layer k if the sample weakly activates existing nodes in the layer ($\forall j, [v_k]_j < \tau_{ad}$) and no collision occurs; see fig. 1(D.1).
2. *Collision*. A sample activates one or more neurons strongly i.e. $\exists j, k, [v_k]_j > \tau_{act}$; the earliest layer where collision occurs is denoted l_c. Such sample is integrated into l_c, thus very similar samples are *selectively clustered*. See fig. 1(D.2).

3. *Filtering into deeper layer* occurs when neither of the above occurs (no strong activation, some moderate activations). Such a sample has features loosely similar to previously seen samples, but we need to filter them further to distinguish its finer features.

Layer 1 construction. To initialize, let $N_1 = (\eta_1^{<1>})$ and $\alpha_1 = (y_1^{<1>})$ where $\eta_1^{<1>} \leftarrow x^{(1)}$ and $y_1^{<1>} \leftarrow y^{(1)}$. Let τ_{ad}, τ_{act} be the *admission threshold* and *activation threshold* respectively. We typically set $\tau_{ad} = 0.1, \tau_{act} = 0.9$. We extend the layer to tuples $N_1 = (\eta_1^{<1>}, \dots, \eta_1^{<n_1>})$ and $\alpha_1 = (y_1^{<1>}, \dots, y_1^{<n_1>})$ through sample-collection function in the pseudo code 1. To do this, take a new sample $(x^{(k)}, y^{(k)}), k > 2$ and we *check* N_1 *activation*, i.e. let $v_1^{(k)}$ be the activation of current layer by this new sample, i.e. $[v_1^{(k)}]_j = \sigma_{dsa}\big(||x^{(k)} - \eta_1^{<j>}||\big)$ for all $\eta_1^{<j>} \in N_1$. Then, either: (1) new sample is *admitted to* N_1 as a new distinct node/neuron. If for all j such that $[v_1^{(2)}]_j < \tau_{ad}$, then $N_1 \rightarrow concat(N_1, x^{(k)})$ and $\alpha_1 \rightarrow concat(\alpha_1, y^{(k)})$. (2) *collision* occurs, when there exists j such that $\tau_{act} < [v_1^{(k)}]_j < 1$, thus sample is admitted via collision resolution mechanism. Exclusively for layer 1, sample will simply be admitted into the layer for selective clustering. New sample causing $[v_1^{(k)}]_j = 1$ collision is unresolvable; see appendix on ill-defined datasets (3) or new sample is *filtered to deeper layers*, when neither occurs. Finally, we complete the iteration over all training data, $N_1 = (\eta_1^{<k>} : k = 1, \dots, n_1)$ and $\alpha_1 = (y^{<k>} : k = 1, \dots, n_1)$.

Lemma 1 *First layer of SQANN achieves arbitrary accuracy on training data subset* $N_1 \times \alpha_1$.

Proof: Let $N_1 = (x^{<1>} = x^{(1)}, x^{<2>}, \dots, x^{<n_1>})$; note that $x^{<k>}$ is not necessarily $x^{(k)}$ except for $k = 1$ for initialization. To prove the lemma, take a sample $(x, y) \in N_1 \times \alpha_1$. Then we must have $x = x^{<j>}, y = \alpha_1^{<j>}$ for some $j = 1, \dots, n_1$. Since $[v_1]_{j'} = \sigma_{dsa}(||x - \eta_1^{<j'>}||)$ and $\eta_1^{<j'>} = x^{<j'>}$ for all $j' = 1, \dots, n_1$, we get exactly $[v_1]_j = 1$. Furthermore, for other $i \neq j$, we have $[v_1]_i < 1$ due to the admission conditions (1) and (2) used during *check* N_1 *process*. Finally, computing $y = \alpha_1^{<j>}$ where $j = argmax_{j'}[v_1]_{j'}$, we retrieve the exact value. □

At this point, it may be clearer to readers how SQANN is constructed. In short, for each layer, *representative activations* become the neurons of the layer. In layer 1, representatives activations are the samples themselves. In deeper layers, they are activations propagated to the layer.

Layer k construction. Layer k construction is similar to layer 1 construction, except collision could occur at any layer $l_c \leq k$ (next paragraph). Assume every layer $l \in \Lambda = \{1, \dots, k-1\}$ have been constructed using $X_{i \in \Lambda} \subseteq X$. Assume there are still unused data samples i.e. $U = X \setminus \{\bigcup_{i=1}^{k-1} X_i\}$ is non-empty, obtained from samples that have been *filtered to deeper layers*. Let $U = \{u^{<1>}, u^{<2>}, \dots\}$ after re-labelling the indices, with corresponding output values $\{y^{<1>}, \dots\}$. Initialize by first checking $v_k^{<1>}$, the activation of $u^{<1>}$ at layer k for collision; if collision occurs, see next paragraph, otherwise, set $N_k = (\eta_k^{<1>}), \alpha_k = (y_k^{<1>})$ i.e. $\eta_k^{<1>} \leftarrow v_k^{<1>}$. Similar to layer 1 construction, perform *check* N_k *activation* on $(u^{<i>}, y^{<i>})$ for $i > 1$ by computing activation $v_k^{<i>}$ and checking it against the existing nodes. One of the three cases occur (1) admission, when $[v_k^{<i>}]_j < \tau_{ad}$ for all $j = 1, \dots, n_k$ and no collision (2) collision, when there exists index j at a *collided layer* $l_c \leq k$ such that $[v_{l_c}^{<i>}]_j > \tau_{act}$ or (3) otherwise. If (1) occurs, the activation $v_k^{<i>}$ is added as a new neuron to the layer, $N_k \rightarrow concat(N_k, v_k^{<i>})$. If (3) occurs, filter the data sample for deeper layer. Assuming no collision, the process is repeated for the next unused data sample $u^{<i>}$ until all remaining data samples are checked. Once done, repeat the process for layer $k+1$ construction.

Suppose collision happens when we check $u^{<m>}$ at layer l_c, we use the collision resolution mechanism. We destroy all layers $l > l_c$ and put the collided sample into l_c, i.e. $N_{l_c} \rightarrow concat(N_{l_c}, v_k^{<m>})$ and $\alpha_{l_c} \rightarrow concat(\alpha_{l_c}, y_k^{<m>})$ (push-node in the pseudo code). No layer will be destroyed if $l_c = k$, the current layer. The data samples used in each destroyed layer are returned to the list of unused samples in the same order they have been used during the construction (return-index() in the pseudo code); we refer to this as *order integrity*. Once the colliding sample is added to l_c, effectively, the strong activation in this layer is now overshadowed by maximum activation (*selective clustering* in action), since the exact neuron is now included as the representative of itself and its locality. This is possibly a practically inefficient process, since we tear down intermediate layers, but we only prioritize the completion of the construction for now.

Computing output via SQANN propagation (prediction). Let an input be x. The output $y = SQANN(x)$ is computed by propagating and processing signals through the layers; fig. 1(B). Then $[v_1]_j = \sigma_{dsa}\big(||x - \eta_1^{<j>}||\big)$. If there exists j such that $v_1^{<j>} > \tau_{act}$, then set $y = \alpha_1^{<j>}$ where $j = argmax_{j'}[v_1]_{j'}$. Otherwise, for subsequent layer k, recursively compute $[v_k]_j = \sigma_{dsa}\big(||v_{k-1} - \eta_k^{<j>}||\big)$ for all $j = 1, \ldots, n_k$. If there exist j, k such that $[v_k]_j > \tau_{act}$, then $y = \alpha_k^{<j>}$ where $j = argmax_{j'}[v_k]_{j'}$. If such layer k is not found, we have to perform interpolation.

Interpolations can be done in many different ways, and this paper implements a simple interpolation using values from the two most strongly activated neurons. Suppose $V_1 = [v_m]_i$ and $V_2 = [v_n]_j$ are the two most activated neurons, then the interpolated value can be, for example, $y = \frac{V_1[\alpha_m]_i + V_2[\alpha_n]_j}{V_1 + V_2}$. The form of interpolation can be adjusted according to the knowledge we have on the dataset, e.g. we can use TNN with high a if we know the dataset is locally constant.

Each training sample x admitted to layer N_k leaves a fingerprint, in the sense that it has a collection of activations $\{v_l | l = 1, \ldots, k\}$ as it is SQANN-propagated through the neural network. This collection is unique amongst training samples, especially because of $[v_k]_j = 1$ where j is an index within layer k it is admitted into. Furthermore, locality is preserved to the extent that if $x' \approx x$, then activation $[v'_k]_j \approx 1$ is around the peak and thus, by SQANN propagation, due to argmax, it is likely we retrieve y, the ground-truth value corresponding to x. For now, we only use argmax, but more subtle adjustment can be done to obtain $y' \approx y$ but $y' \neq y$ for x' by incorporating information about the manifold at that locality, if such knowledge is available.

Completing construction. Due to collisions, readers might wonder if the construction will complete at all. During collision, layers are torn down and reconstructed. Suppose during layer k construction, collision occurs at layer c for $c < k$. Upon reconstruction back to layer k, layer c may be torn down again in the next collisions. Is it possible that collision occurs infinitely cyclically? The following proposition addresses the concern through *order integrity* previously mentioned.

Proposition 4 *SQANN construction completes with high probability $p \approx 1$. See appendix for (1) sketch of proof and a required assumption (2) stronger assumption needed for $p = 1$.*

Arbitrarily high accuracy on training dataset D is relatively simple to prove in the following theorem: roughly for each $(x, y) \in D$, there exist l, k such that x maximally activates the node $\eta_l^{<k>}$, thus the correct y is guaranteed to be fetched from α_l. Catastrophic forgetting resistance is proven similarly: when new samples are used for training, previous samples are not forgotten since SQANN stores the particular fingerprint $\eta_l^{<k>}$ for each sample.

Theorem 2 *Assume SQANN construction is completed. SQANN achieves arbitrarily high accuracy on a training dataset. Furthermore, it is resistant to catastrophic forgetting. Proof: see appendix. Note: Our code provides experimental demonstrations showing zero errors on all training samples.*

SQANN pencil-and-paper example. With $a_1, a_2 = 0.001, 0.5$, $\tau_{ad}, \tau_{act} = 0.1, 0.9$, create SQANN universal approximator for indexed data $X = [x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}] = \begin{bmatrix} 1 & 1.2 & -1 & -1.2 \\ 1.2 & 0.8 & -1 & -1.2 \end{bmatrix}$ and $Y = [y_1, y_2, y_3, y_4] = [1, 1, 0, 0]$. See follow up questions and full solutions in the appendix.

3.2 Experiment to test Generalizability of SQANN

Test datasets with increasing spread from training distribution. The accuracy of SQANN on high-dimensional dataset outside the training dataset is harder to formalize in theorems. Furthermore, real life data is often noisy and possibly not regularly structured. We avoid making any related statements for SQANN for now. We instead provide empirical results on test datasets that are *similar* to the training dataset, to the extent that each point in the test dataset is a training sample perturbed by uniform random values of increasing magnitude. We refer to the noise magnitude as the *test data spread*. Fig. 4(A,A.2) show four domains X with different test data spread. Test dataset that has larger test data spread contains data samples that are noisier and further away from the training data points. Fig. 4(B,B.2) show that SQANN naturally performs better with smaller test data spread. As the test data spread increases, larger errors are observed. Likewise, smaller spread means smaller N_{interp}, i.e. fewer data samples fail to activate neurons in SQANN strongly. For all, training errors are 0 *as expected from theorem 2*.

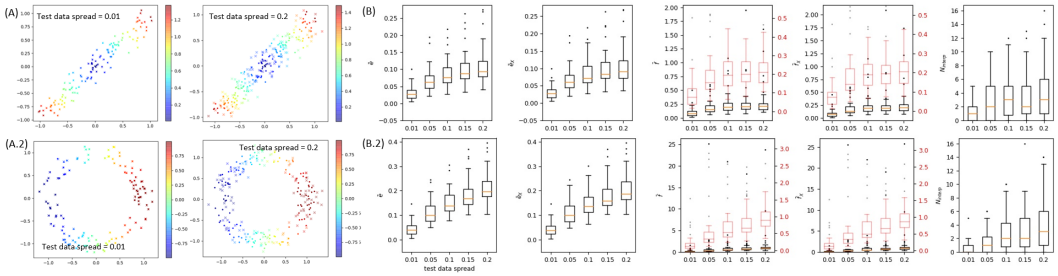

Figure 4: (A) Training/test (circles/x marks) data for demonstration. Smaller/larger test data spread means test samples are closer/further to/from training samples. (B) Boxplots for data whose distributions are similar to (A). Column 1(3): (fractional) errors on test data samples. Column 2(4), (fractional) errors on test data samples excluding interpolated samples. Column 5: no. of data samples whose predicted values are interpolated. (B.2) Similar to top, but for (A.2).

Classification and visualization of SQANN's special interpretability features. We show the use of SQANN for a simple classification problem in fig. 1(C). The ring outside is labelled 0.5, while the ring inside 1.0. With activation parameters $a_1 = a_2 = 1$, we achieve zero error not only for training dataset (to be expected from theorem 2) but also on test dataset. A special feature in SQANN is its ability to tell the user which data samples fail to activate any neurons strongly; such samples' output must be interpolated (see SQANN propagation). In fig. 1(C) right, points marked with red open circles need interpolation. Each such point is interpolated using two training samples whose "fingerprint" neurons are most strongly excited. These two samples are shown as the two points directly linked by colored straight lines to the open circle. This is possible because SQANN systematically stores indices of training data samples within respective layers. The list of indices organized by layers can even be explicitly printed e.g. see SQANN.ipynb, supp. materials.

SQANN is tested for regression on Boston Housing and Diabetes Datasets to demonstrate its generalizability to unseen/test samples, as the following (simplified here): (1) A small subset of samples $D \subseteq \mathcal{D}$ (the first 20% of full dataset $\mathcal{D}$) is used to train SQANN and 9 other regression methods. (2) Mean Squared Errors (MSE) values are measured on unused data D_{test} on all 10 models; we expect large errors on some test samples. (3) SQANN's activations are used to collect samples with large absolute errors $e_\tau(x) = |SQANN(x) - y_0| > \tau$ and we treat them as out-of-distribution (OOD) samples. These samples are considered as new distinct samples to be integrated into D as the new training dataset D'. (4) Train the 10 models, now with D'. (5) Then MSE is measured again on D_{test} (yes, there will be partial overfitting). From Boston dataset: for $\tau = 5$, SQANN MSE improved from 9.90 to 3.08. Decision tree improves the most with D' (7.36 to 2.07); see table 1. For more details and diabetes dataset, see the appendix.

4 Conclusion, Limitation and Future Direction

Limitations and future directions. **TNN** is clearly limited in regards to its application to multi-dimensional input data. However, it might find interesting uses in time series, such as ECG (Electrocardiogram). ECG signals can be approximated point-wise, though it might be better to have a noise model to prevent overfitting. In particular, PQRST segments from ECG can be mapped to specific neurons within TNN, giving some neurons specific meaning and thus interpretability. **SQANN** limitation and possible future development currently include 1) simple sequential drawing of samples that may result in the imbalance of layer size. In the future, more sophisticated ordering of training samples can be used so that layers are constructed with meaningful and purposeful arrangement e.g. deeper layers can be purposefully reserved for rare cases; more research on this is necessary to optimize the results 2) layer destruction during the treatment of collision cases might be an inefficient mechanism, which can be improved in the future. In conclusion, we have proposed TNN and SQANN, two interpretable NNs for universal approximation designed to (1) be resistant to catastrophic forgetting (2) have provably high accuracy on training datasets and (3) be directly interpretable via neurons' activation patterns.

Table 1: Comparing MSE on different regression methods for Boston Housing dataset. Row o., or original, shows MSE obtained from models trained on D. Row eT shows MSE from models trained on D' with $\tau = T$. All are evaluated on D_{test}.

	Lin	Ridge	Lasso	LSVR	NuSVR	SVR	DTree	kneigh	MLP	SQANN
o.	36.45	7.990	9.834	8.355	8.833	8.712	7.356	7.393	12.77	9.898
e5	5.139	5.135	7.068	5.950	6.295	6.068	5.026	4.798	3.481	7.998
e2	4.993	5.028	7.882	5.832	6.121	5.895	**2.072**	3.025	3.846	**3.076**

Note on table 1. The entries in the header denote the models available in scikit-learn (Pedregosa et al. (2011b)): Lin, Ridge and Lasso are the linear models: linear , Ridge (linear least square with L2 regularization), Lasso (linear with L1) respectively; the Support Vector Regression models: LinSVR, NuSVR, SVR are respectively linear SVR, ν-SVR and ϵ-SVR. DTree: Decision tree; kneigh: k neighbours are selected from the best $k = 2, 3, ..., 16$, MLP: multi-layer perceptrons, or the fully-connected neural network, with 2 layers, each layer having 64 neurons each trained for a max of 12000 iterations (convergence is attained for both). For SQANN, the initial model trained on D is kept after SQANN' is trained on D'. Thus, we can choose results based on the strength of activations between SQANN and SQANN'.

For Boston Housing Dataset $\tau = 5$ (i.e. e5 of table 1), using SQANN we integrated 211 samples from the test dataset into training dataset, so $|D'| = 311$. Overall, 0.615 of the whole $\mathcal{D}$ is used for new training. For $\tau = 2$, i.e. e2 of table 1, using SQANN we integrated 319 samples, so $|D'| = 419$ i.e. 0.828 of the whole $\mathcal{D}$ is used for new training. With $\tau = 2$, regression performance of SQANN improves greatly compared to other methods, except for decision tree regression. We have thus also seen that SQANN can be used to perform sample selections for data that appear to be out of distribution; this has improved decision tree performance greatly. The performance of other models have improved reasonably too, especially MLP. For MLP, however, the randomness used to achieve convergence to some local minima might have led it to explore other minima; hence we get slightly decreased performance for e2 compared to e5.

```
Main SQANN loop:
function fit_data(X,Y){
  l_now=1 # layer now
  while True {
    ssig, collision = sample_collection(X, Y, l_now)
    if ssig is 'no more data':
      break
    else if ssig is 'collision':
      l_c = collision['collided_layer']
      for l_j from l_c+1 to l_now+1:
        return_index(l_j)
      kp = collision['perpetrator_index']
      push_node(kp,X[kp,:],Y[kp],l_c)
      l_now = l_c}
  l_now+=1 }
```

```
function sample_collection(X,Y,layer){
  i=unused_indices[0]
  x=X[i,:]
  x, collision = forward_cons(x,layer−1)
  ssig, collision = check_signal(collision)
  nodes, node_values = new_nodes(x,Y[i])
  remove_index(i,layer)
  for i in unused_indices{
    x=X[i,:]
    x, collision = forward_cons(x,layer−1)
    ssig, collision = check_signal(collision)
    act=activate(x,nodes)
    if all(act<admission_threshold){
      update_nodes(x,Y[i],nodes,node_values)
      remove_index(i,layer)}
  }return ssig, collision}
```

Pseudo code 1: Pseudo code for the construction of SQANN. The function *activate* corresponds to equation 5. See appendix for mapping to python code.

ETHICS STATEMENTS

This paper introduces function approximators with novel properties. It is purely mathematical and algorithmic. No specific ethical issues are present. The ethical context depends only on the dataset but in this paper, only common public datasets have been used.

REPRODUCIBILITY STATEMENTS

All codes are available in the supp. materials (to be released to public repository in case of acceptance). Results are easily reproducible even with without random number seeding since data

distributions are sufficiently controlled. Jupyter notebook for our particular results are also present. Proofs are all included in the appendix, with sketch of proof and additional statement of assumptions where applicable.

ACKNOWLEDGMENTS

Anonymous for now.

REFERENCES

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82 – 115, 2020. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2019.12.012. URL http://www.sciencedirect.com/science/article/pii/S1566253519308103.

C. L. Philip Chen, Zhulin Liu, and Shuang Feng. Universal approximation capability of broad learning system and its structural variations. *IEEE Transactions on Neural Networks and Learning Systems*, 30(4):1191–1204, 2019. doi: 10.1109/TNNLS.2018.2866622.

C. K. Chui, Xin Li, and H. N. Mhaskar. Neural networks for localized approximation. *Mathematics of Computation*, 63(208):607–623, 1994. ISSN 00255718, 10886842. URL http://www.jstor.org/stable/2153285.

C. K. Chui, Xin Li, and H. N. Mhaskar. Limitations of the approximation capabilities of neural networks with one hidden layer. *Advances in Computational Mathematics*, 5(1):233–243, Dec 1996. ISSN 1572-9044. doi: 10.1007/BF02124745. URL https://doi.org/10.1007/BF02124745.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989. ISSN 1435-568X. doi: 10.1007/BF02551274. URL https://doi.org/10.1007/BF02551274.

L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 80–89, 2018.

Boris Hanin. Universal function approximation by deep neural nets with bounded width and relu activations. *Mathematics*, 7(10), 2019. ISSN 2227-7390. doi: 10.3390/math7100992. URL https://www.mdpi.com/2227-7390/7/10/992.

Jesse Johnson. Deep, skinny neural networks are not universal approximators. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGgSsAcFQ.

Patrick Kidger and Terry Lyons. Universal approximation with deep narrow networks, 2020. URL https://openreview.net/forum?id=B1xGGTEtDH.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. ISSN 0027-8424. doi: 10.1073/pnas.1611835114. URL https://www.pnas.org/content/114/13/3521.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, Mar 2021. ISSN 2522-5839. doi: 10.1038/s42256-021-00302-5. URL https://doi.org/10.1038/s42256-021-00302-5.

H. N. Mhaskar. Neural networks for localized approximation of real functions. In *Neural Networks for Signal Processing III - Proceedings of the 1993 IEEE-SP Workshop*, pp. 190–196, 1993a.

H. N. Mhaskar. Approximation properties of a multilayered feedforward artificial neural network. *Advances in Computational Mathematics*, 1(1):61–80, 1993b. ISSN 1572-9044. doi: 10.1007/BF02070821. URL https://doi.org/10.1007/BF02070821.

H. N. Mhaskar. Neural networks for optimal approximation of smooth and analytic functions. *Neural Computation*, 8(1):164–177, 1996.

H.N Mhaskar and Charles A Micchelli. Approximation by superposition of sigmoidal and radial basis functions. *Advances in Applied Mathematics*, 13(3):350 – 373, 1992. ISSN 0196-8858. doi: https://doi.org/10.1016/0196-8858(92)90016-P. URL https://www.sciencedirect.com/science/article/pii/019688589290016P.

Sejun Park, Chulhee Yun, Jaeho Lee, and Jinwoo Shin. Minimum width for universal approximation. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=O-XJwyoIF-k.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011a. URL https://scikit-learn.org/stable/modules/neighbors.html.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011b.

Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta Numerica*, 1999.

M. A. Sartori and P. J. Antsaklis. A simple method to derive bounds on the size and to train multilayer neural networks. *IEEE Transactions on Neural Networks*, 2(4):467–471, 1991.

Erico Tjoa and Cuntai Guan. A survey on explainable artificial intelligence (xai): Toward medical xai. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2020. doi: 10.1109/TNNLS.2020.3027314.

Sarah Wiegreffe and Ana Marasović. Teach me to explain: A review of datasets for explainable nlp. *ArXiv*, abs/2102.12060, 2021.

Guang Yang, Feng Pan, and Wen-Biao Gan. Stably maintained dendritic spines are associated with lifelong memories. *Nature*, 462(7275):920–924, Dec 2009. ISSN 1476-4687. doi: 10.1038/nature08577. URL https://pubmed.ncbi.nlm.nih.gov/19946265.19946265[pmid].