Atlas: Universal Function Approximator For Memory Retention

Anonymous Author(s)
Affiliation
Address
email

Abstract

Artificial neural networks (ANNs), despite their universal function approximation capability and practical success, are subject to catastrophic forgetting. Catastrophic forgetting refers to the abrupt unlearning of a previous task when a new task is learned. It is an emergent phenomenon that plagues ANNs and hinders continual learning. Existing universal function approximation theorems for ANNs guarantee function approximation ability, but seldom touch on the model details and do not predict catastrophic forgetting. This paper presents a novel universal approximation theorem for multi-variable functions using only single-variable functions and exponential functions. Furthermore, we present *Atlas*—a novel ANN architecture based on the exponential approximation theorem and B-splines. It is shown that Atlas is a universal function approximator capable of memory retention and, therefore, continual learning. The memory retention of Atlas is imperfect, with some off-target effects during continual learning, but it is well-behaved and predictable. An efficient implementation of Atlas is provided. Experiments are conducted to evaluate both the function approximation and memory retention capabilities of Atlas.

1 Introduction

Universal function approximation theorems are a cornerstone of machine learning, and prove that artificial neural networks (ANNs) can approximate any given continuous target function with arbitrarily small error [6, 7, 11]. The theorems do not specify how to find a given ANN architecture and parameters with sufficient performance for problems in practice. Another problem that arises in practice is catastrophic forgetting [3, 9, 15], where an ANN would learn a new task and the subsequent parameter updates would interfere with the model's performance on previously learned tasks. Catastrophic forgetting is also called catastrophic interference [13]. Catastrophic forgetting is like learning to pick up a cup, but simultaneously forgetting how to breathe.

If an ANN cannot effectively learn many tasks, it has limited utility in the context of continual learning [4, 8]. ANNs have other practical problems, such as vanishing or exploding gradients [5], which make training unstable. It is also not obvious how to increase the size of a trained ANN to better fit to data without damaging model performance and having to retrain the model on already seen data.

This paper introduces Atlas—a novel universal function approximator based on B-splines that has intrinsic memory retention. Atlas has well-behaved parameter gradients that do not vanish or explode. Atlas has methods to expand model capacity in a systematic way without loosing the previously learned information. The accompanying representation and universal approximation theorems are also provided.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

2 Relevant Studies

It is conjectured that overlapping representations in ANNs lead to catastrophic forgetting [8]. Catastrophic forgetting occurs when parameters necessary/needed for one task change while training to meet the objectives of another task [10, 14]. Strategies to mitigate catastrophic forgetting can include shielding important weights from changes, or regularisation to keep weights close to their initial values. The least desirable option is retraining a model over all tasks, again. Data augmentation approaches like rehearsal and pseudo-rehearsal have also been employed [15].

Pi-sigma neural networks use nodes that compute products instead of sums [17]. Pi-sigma neural networks reportedly have potential utility in building more efficient and expressive models than the standard ANNs. Atlas can similarly compute products, and is more easily trained than the Pi-sigma neural networks.

B-splines, which form the basis of Atlas, have been applied for machine learning [2]. Scardapane et al. [16] investigated trainable activation functions parameterised by splines. Uniform cubic B-splines have basis functions that are translates of one another [1]. Uniform cubic B-splines have been tested for memory retention, and Atlas can be considered as an improvement on existing spline additive models [18].

3 Exponential Representation Theorem

Any continuous multi-variable function on a compact space can be uniformly approximated with multi-variable polynomials by the Stone-Weierstrass Theorem. Let $\mathcal{I}$ denote an index set of tuples of natural numbers including zero such that $i_j \in \mathbb{N}^0$ for all $j \in \mathbb{N}$ with $i = (i_1, .., i_n) \in \mathcal{I}$ and $a_i \in \mathbb{R}$. Multi-variable polynomials can be represented as:

$$y(\vec{\mathbf{x}}) = y(x_1, .., x_n) = \sum_{i \in \mathcal{I}} a_i x_1^{i_1} x_2^{i_2} ... x_n^{i_n} = \sum_{i \in \mathcal{I}} a_i \Pi_{j=1}^{n} x_j^{i_j}$$

Each monomial term $a_i \Pi_{j=1}^{n} x_j^{i_j}$ is a product of single-variable functions in each variable. It is desirable to rewrite products as sums. The product of single-variable functions can be rewritten as a sum using exponentials and logarithms.

Lemma 1. *For any $a_i \in \mathbb{R}$, there exists $\gamma_i > 0$ and $\beta_i > 0$, such that: $a_i = \gamma_i - \beta_i$*

Theorem 1 (Exponential representation theorem)**.** *Any multi-variable polynomial function $y(\vec{\mathbf{x}})$ of n variables over the positive orthant, can be exactly represented by continuous single-variable functions $g_{i,j}(x_j)$ and $h_{i,j}(x_j)$ in the form:*

$$y(\vec{\mathbf{x}}) = \sum_{i \in \mathcal{I}} \exp\big(\Sigma_{j=1}^{n} g_{i,j}(x_j)\big) - \exp\big(\Sigma_{j=1}^{n} h_{i,j}(x_j)\big)$$

Proof. Consider any monomial term $a_i \Pi_{j=1}^{n} x_j^{i_j}$ with $a_i \in \mathbb{R}$, then by Lemma 1 there exist strictly positive numbers $\gamma_i > 0$ and $\beta_i > 0$, such that:

$$\begin{aligned} a_i \Pi_{j=1}^{n} x_j^{i_j} &= \gamma_i \Pi_{j=1}^{n} x_j^{i_j} - \beta_i \Pi_{j=1}^{n} x_j^{i_j} \\ &= \exp\Big(\log\Big(\gamma_i \Pi_{j=1}^{n} x_j^{i_j}\Big)\Big) - \exp\Big(\log\Big(\beta_i \Pi_{j=1}^{n} x_j^{i_j}\Big)\Big) \\ &= \exp\Big(\log(\gamma_i) + \Sigma_{j=1}^{n} \log\Big(x_j^{i_j}\Big)\Big) - \exp\Big(\log(\beta_i) + \Sigma_{j=1}^{n} \log\Big(x_j^{i_j}\Big)\Big) \end{aligned}$$

The argument of each exponential function is a sum of single-variable functions and constants. Without loss of generality, a set of single-variable functions can be defined such that:

$$a_i \Pi_{j=1}^{n} x_j^{i_j} = \exp\big(\Sigma_{j=1}^{n} g_{i,j}(x_j)\big) - \exp\big(\Sigma_{j=1}^{n} h_{i,j}(x_j)\big)$$

Since this holds for any $a_i \Pi_{j=1}^{n} x_j^{i_j}$ and all $i \in \mathcal{I}$, it follows that:

$$y(\vec{\mathbf{x}}) = \sum_{i \in \mathcal{I}} \exp\big(\Sigma_{j=1}^{n} g_{i,j}(x_j)\big) - \exp\big(\Sigma_{j=1}^{n} h_{i,j}(x_j)\big)$$

□

This result is fundamental to the paper. Since every continuous function can be approximated with multi-variable polynomials, it follows that every continuous function can be approximated with positive and negative exponential functions. Some consideration must be given to single-variable function approximators, since it is the main building block for the representation theorem. A lot of design and thought went into choosing a single-variable function approximator.

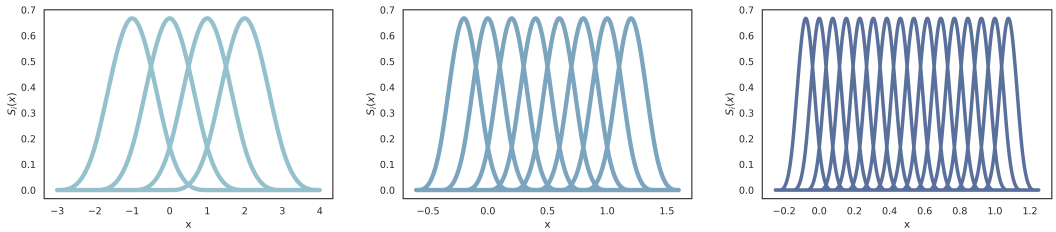
4 Single-Variable Function Approximation in Atlas

Each single-variable function in Atlas is approximated with uniform cubic B-spline basis functions. The choice was made to use uniform cubic B-splines due to their excellent performance and robustness to catastrophic forgetting. Each basis function is multiplied by a parameter and summed together. The number of basis functions is typically fixed and remains unchanged. With uniform B-splines, each basis function is scaled so that the unit interval is uniformly partitioned as in Figure 1.

Definition 1 (ρ-density B-spline function)**.** A ρ-density B-spline function is a uniform cubic B-spline function with $2^{\rho+2}$ basis functions:

$$f(x) = \sum_{i=1}^{2^{\rho+2}} \theta_i S_i(x) = \sum_{i=1}^{2^{\rho+2}} \theta_i S(w_i x + b_i) = \sum_{i=1}^{2^{\rho+2}} \theta_i S((2^{\rho+2} - 3)x + 4 - i)$$

The limitation of uniform B-spline basis functions is that increasing the number of basis functions would alter each basis function, which means any previously learned parameters would either be discarded or modified in some undetermined way. If one instead used the familiar Fourier basis functions, then one could add more basis functions, but initialise their coefficients to be zero. Recall that adding zero to a function does not change it, leaving the overall model unchanged. In order to achieve something similar with B-spline basis functions, instead of using a single density, we propose adding together uniform B-spline basis functions with different densities, as illustrated in Figure 1.



(a) $\rho = 0$ (b) $\rho = 1$ (c) $\rho = 2$

Figure 1: Doubling densities of basis functions.

The minimum number of cubic B-spline basis functions is four; we define this as a density $\rho = 0$. A density of $\rho = 1$ is (doubled) eight uniform cubic B-spline basis functions. A density of $\rho = 2$ is sixteen uniform cubic B-spline basis functions, and so forth. Atlas uses a double-summation, summing together all cubic B-spline basis functions, for each density up to some maximum fixed density r.

Definition 2 (mixed-density B-spline function)**.** A mixed-density B-spline function is a single-variable function approximator that is obtained by summing together different ρ-density B-spline functions. Only the maximum ρ-density B-spline function has trainable parameters, the others are constant. Mixed-density B-spline functions are of the form:

$$f(x) = \sum_{\rho=0}^{r} \sum_{i=1}^{2^{\rho+2}} \theta_{\rho,i} S_{\rho,i}(x)$$

Analytically, we can choose all the new parameters $\theta_{r+1,i} = 0,\ \forall i \in \mathbf{N}$ such that:

$$f(x) = \sum_{\rho=0}^{r} \sum_{i=1}^{2^{\rho+2}} \theta_{\rho,i} S_{\rho,i}(x) = \sum_{\rho=0}^{r+1} \sum_{i=1}^{2^{\rho+2}} \theta_{\rho,i} S_{\rho,i}(x)$$

The last thing to note is that only the parameters for the largest specified density are trainable, in contrast to smaller density parameters that are fixed constants. It is possible to create a minimal model with $r = 0$ initialised at zero, and train the model until convergence. Then one can create a new model with $r = 1$, by subsuming the previous model's parameters. The larger and more expressive model can be trained up to convergence. This process of training and expansion can be continued indefinitely and is shown in Figure 2. Every single-variable function in Atlas is a mixed-density B-spline function.

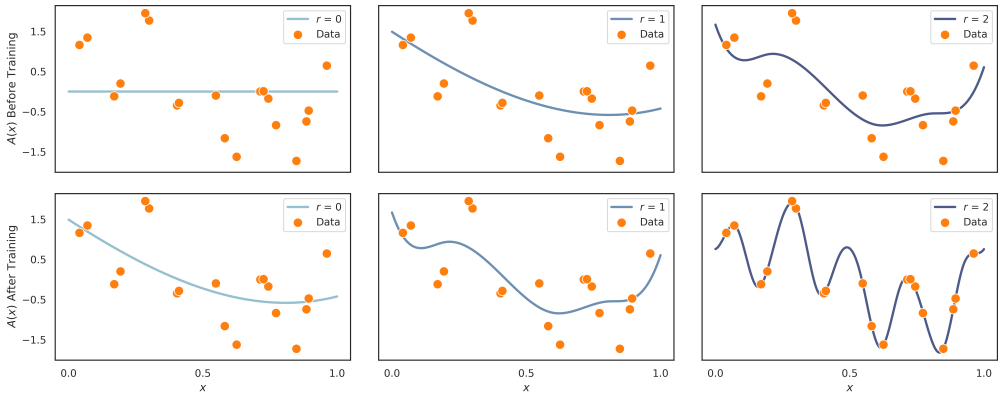


Figure 2: Doubling densities of basis functions before and after training.

5 Atlas

Atlas is named for carrying the burden of all it must remember, after the Titan god Atlas in Greek mythology who was tasked with holding the weight of the world.

Theorem 2 (Atlas representation theorem)**.** *Any multi-variable polynomial $y(\vec{\mathbf{x}})$ of n variables over the positive orthant, can be exactly represented by continuous single-variable functions $f_j(x_j)$, $g_{i,j}(x_j)$, and $h_{i,j}(x_j)$ in the form:*

$$y(\vec{\mathbf{x}}) = \sum_{j=1}^{n} f_j(x_j) + \sum_{k=1}^{\infty} \frac{1}{k^2} \exp\left(\Sigma_{j=1}^{n} g_{k,j}(x_j)\right) - \frac{1}{k^2} \exp\left(\Sigma_{j=1}^{n} h_{k,j}(x_j)\right)$$

Definition 3 (Atlas)**.** Atlas is a function approximator of n variables, with mixed-density B-spline functions $f_j(x_j)$, $g_{i,j}(x_j)$, and $h_{i,j}(x_j)$ in the form:

$$A(\vec{\mathbf{x}}) := \sum_{j=1}^{n} f_j(x_j) + \sum_{k=1}^{M} \frac{1}{k^2} \exp\left(\Sigma_{j=1}^{n} g_{k,j}(x_j)\right) - \frac{1}{k^2} \exp\left(\Sigma_{j=1}^{n} h_{k,j}(x_j)\right)$$

Atlas is equivalently given by the compact notation:

$$A(\vec{\mathbf{x}}) := F(\vec{\mathbf{x}}) + \sum_{k=1}^{M} \frac{1}{k^2} \exp(G_k(\vec{\mathbf{x}})) - \frac{1}{k^2} \exp(H_k(\vec{\mathbf{x}}))$$

The number of exponential terms can be increased without changing the output of the model. We can choose to initialise $G_{M+1}(\vec{\mathbf{x}}) = 0$ and $H_{M+1}(\vec{\mathbf{x}}) = 0$, such that the model capacity can be increased without changing the output of the model:

$$\begin{aligned}
A_{M+1}(\vec{\mathbf{x}}) &= \sum_{k=1}^{M+1} \frac{1}{k^2} \exp(G_k(\vec{\mathbf{x}})) - \frac{1}{k^2} \exp(H_k(\vec{\mathbf{x}})) \\
&= \frac{1}{(M+1)^2} \exp(G_{M+1}(\vec{\mathbf{x}})) - \frac{1}{(M+1)^2} \exp(H_{M+1}(\vec{\mathbf{x}})) + A(\vec{\mathbf{x}}) \\
&= \frac{1}{(M+1)^2} \exp(0) - \frac{1}{(M+1)^2} \exp(0) + A(\vec{\mathbf{x}}) \\
&= A(\vec{\mathbf{x}})
\end{aligned}$$

Atlas is a universal function approximator with robust memory retention. It possesses three properties atypical of most universal function approximators:

1. The activity within Atlas is sparse – most neural units are zero and inactive.
2. The gradient vector with respect to trainable parameters is bounded regardless of the size and capacity of the model, so training is numerically stable for many possible training hyper-parameters.
3. Inputs that are sufficiently far from each other have orthogonal representations.

Property 1 (Sparsity)**.** *For any $\vec{\mathbf{x}} \in D(A) \subset R^n$ and bounded trainable parameters θ_i with index set Θ, the gradient vector of trainable parameters (for Atlas) is sparse:*

$$\left\| \vec{\boldsymbol{\nabla}}_{\vec{\theta}} A(\vec{\mathbf{x}}) \right\|_0 = \sum_{i \in \Theta} d_{Hamming} \left(\frac{\partial A}{\partial \theta_i}(\vec{\mathbf{x}}), 0 \right) \leq 4n(2M+1)$$

Remark. For a fixed number of variables n, the model has a total of $n2^{r+2}(2M+1)$ trainable parameters. The gradient vector has a maximum of $4n(2M+1)$ non-zero entries, which is independent of r. Recall that only the maximum density ($\rho = r$) cubic B-spline function has trainable parameters. The fraction of trainable basis functions that are active is at most 2^{-r}. Sparsity entails efficient implementation, and suggests possible memory retention and robustness to catastrophic forgetting.

Property 2 (Gradient flow attenuation)**.** *For any $\vec{\mathbf{x}} \in D(A) \subset R^n$ and bounded trainable parameters θ_i with index set Θ: if all the mixed-density B-spline functions are bounded, then the gradient vector of trainable parameters for Atlas is bounded:*

$$\left\| \vec{\boldsymbol{\nabla}}_{\vec{\theta}} A(\vec{\mathbf{x}}) \right\|_1 = \sum_{\theta_i \in \Theta} \left| \frac{\partial A}{\partial \theta_i}(\vec{\mathbf{x}}) \right| < U$$

Remark. For a fixed number of variables n, the model has a total of $n2^{r+2}(2M+1)$ trainable parameters. The factor of k^{-2} inside the expression for Atlas is necessary to ensure the sum is convergent in the limit of infinitely many exponential terms $M \to \infty$. Only the maximum density ($\rho = r$) cubic B-spline function has trainable parameters, so that the gradient vector is bounded in the limit of arbitrarily large densities $r \to \infty$. It is worth recalling that at most four basis functions are active for uniform cubic B-spline functions, regardless of the density, but the smaller densities cannot be trainable, otherwise this property does not hold. The gradient vector has bounded L1 norm for any number of basis functions and exponential terms. The bounded gradient vector implies that Atlas is numerically stable during training, regardless of its size or parameter count.

Property 3 (Distal orthogonality)**.** *For any $\vec{\mathbf{x}}, \vec{\mathbf{y}} \in D(A) \subset R^n$ and bounded trainable parameters θ_i, there exists a $\delta > 0$ such that:*

$$\min_{j=1,\ldots,n} \{|x_j - y_j|\} > \delta \implies \langle \vec{\boldsymbol{\nabla}}_{\vec{\theta}} A(\vec{\mathbf{x}}), \vec{\boldsymbol{\nabla}}_{\vec{\theta}} A(\vec{\mathbf{y}}) \rangle = 0$$

Remark. Two points that sufficiently differ in each input variable have orthogonal parameter gradients. It is worth mentioning that the condition resembles a cross-like region in two variables, and planes that intersect in higher dimensions. Distal orthogonality means Atlas is reasonably robust to catastrophic forgetting.

The absolutely convergent series of scale factors k^{-2} was chosen for numerical stability and to ensure the model is absolutely convergent. Another feature is that the series of scale factors also breaks the symmetry that would otherwise exist if all mixed-density B-spline functions were initialised to zero. Initialising all the parameters to be zero is a departure from the conventional approach of random initialisation.

Atlas can be implemented with 1D convolutional layers and dense layers, as outlined in Figure 3. Only some of the dense layers have trainable parameters. The same basis functions are used for each input variable, hence the choice of 1D convolutions with fixed parameters and spline activations to correctly implement mixed-density B-spline functions. Dense core corresponds to the lower ρ-density B-spline function parameters that are not trainable. Dense summation sums together the positive and negative exponential functions with the appropriate scaling factors. It is worth mentioning that an efficient implementation of Atlas converged on using 1D convolutional layers as a method for approximating high-dimensional functions, which potentially suggests a deeper connection to other high-dimensional convolutional models (with small filters).

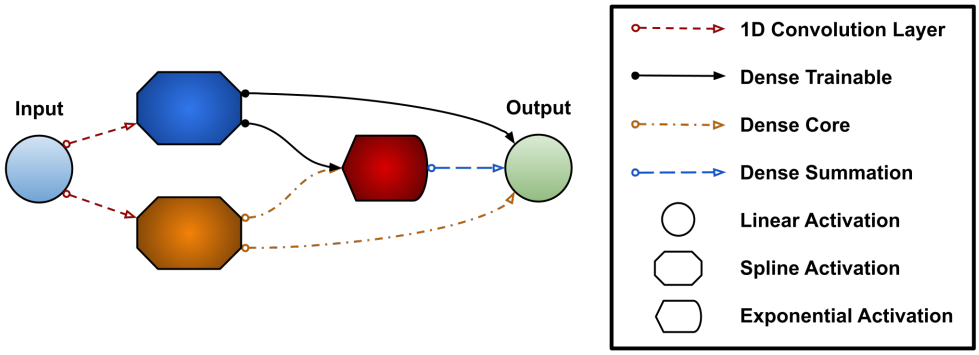


Figure 3: Diagram of an implementation of Atlas using 1D convolution layers and dense layers.

6 Methodology

Multiple experiments were conducted to test the function approximation capability and memory retention properties of Atlas. For the sake of brevity, only the results of Experiment A are presented in this paper. Additional experiments can be found in the supplementary material.

Experiment A was constructed for two reasons: to showcase model training-expansion cycles for a two-variable function; and to demonstrate Atlas memory retention. The swiss-roll target function was chosen because it can be visually analysed, and it is a well-known low-dimensional problem [12], not trivially fit with a standard ANN.

Experimentation was performed with Python and TensorFlow. Experiment A was performed on a personal laptop with a 7th generation i7 Intel processor and took a few hours to finish thirty trials. The loss function chosen for training and evaluation is the mean absolute error (MAE). The training data set and test set in all experiments had 10000 data points, sampled uniformly at random. Gaussian noise with standard deviation 0.1 was added to all training and test data target values. The test set was also used as a validation set to quantify the test error during training. All models were trained with a learning rate of 0.01 with the Adam optimizer. All models and experiments used batch sizes of 100 during training.

To test memory retention, two tasks, presented to Atlas one after the other, were constructed. The details of each task are given below.

Task 1 The training and test sets were sampled uniformly from the Task 1 target function over the domain $[0., 1.]^2$, with Gaussian noise added to the target values. The initial Atlas model was instantiated as a two-variable function that maps to a one-dimensional output, with $r = 0$ and $M = 0$ such that it is a minimally expressive model. The model was evaluated and trained for 30 epochs. After training the Atlas ANN model was expanded using the built-in methods, such that r increases

with one, and M is increased by two: $r' = r + 1$ and $M' = M + 2$. This training-expansion process is repeated four times. The output of the model is presented at the end of each expansion iteration.Where the radius is measured from the centre of the domain $[0.5, 0.5]$, such that the Task 1 target function is given by:

$$r = \sqrt{(x_1 - \frac{1}{2})^2 + (x_2 - \frac{1}{2})^2}$$
$$\theta = \tan^{-1}\left((x - \frac{1}{2})^2, (y - \frac{1}{2})^2\right)$$
$$Y_A = Y_A(x_1, x_2) = \sin(30r + \theta) + 2$$

Task 2 The test sets were sampled uniformly from the Task 2 target function over the domain $[0., 1.]^2$, with Gaussian noise added to the target values. The training sets were sampled uniformly over the domain $[0.45, 0.55]^2$, and target values of zero with added Gaussian noise. All models were trained for 6 epochs. The Task 2 target function is given by:

$$Y'_A(x_1, x_2) = \begin{cases} 0 & 0.45 < x_1 < 0.55, \text{ and } 0.45 < x_2 < 0.55 \\ Y_A(x_1, x_2) & \text{otherwise.} \end{cases}$$

Task 2 effectively tests if a model changes only where new data was presented, with off-target effects leading to larger test MAE.

7 Results

Task 1 The output of the model after training for each expansion iteration is presented in Figure 4. The newly created trainable parameters for each expansion is set to zero, such that the expanded models are equal to the previous model before training, similar to Figure 2.

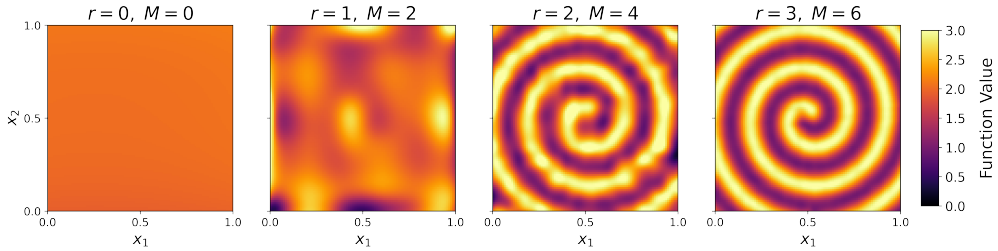


Figure 4: Outputs of the model during successive training and expansion iterations.

The mean training and validation loss curves are presented in Figure 5, with standard deviation shaded in. There is markedly little variation. The initial model converged almost immediately, and practically resembles a constant function as seen in Figure 4. Each expansion iteration is indicated with a vertical line. It is clear that the model was able to better approximate the target function with increased capacity.

Task 2 The final model from Task 1, with $r = 4$ and $M = 8$, was trained on the subset $[0.45, 0.55]^2$. The outputs of the model and the target functions were visualised in Figure 6 with grid-sampled points. The images from the left: Task 1 target function; model output after fully training on Task 1; Task 2 target function; model output after training on Task 2; The absolute difference between the models' first and second output. A model with nearly perfect memory retention would only differ in the small square region $[0.45, 0.55] \times [0.45, 0.55]$ in the centre of the unit square.

It is evident from Figure 6 that there are some off-target effects, and catastrophic forgetting. However, as predicted there are regions that were left untouched. Only a small cross-shaped region was susceptible to off-target effects and catastrophic forgetting.

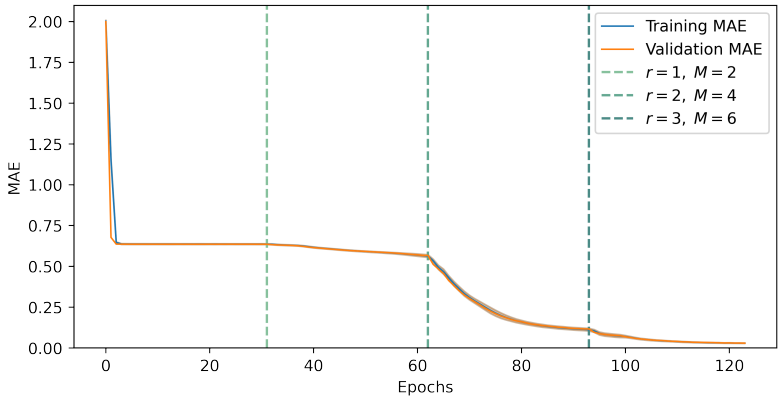


Figure 5: Training and validation MAE during the course of training on Task 1

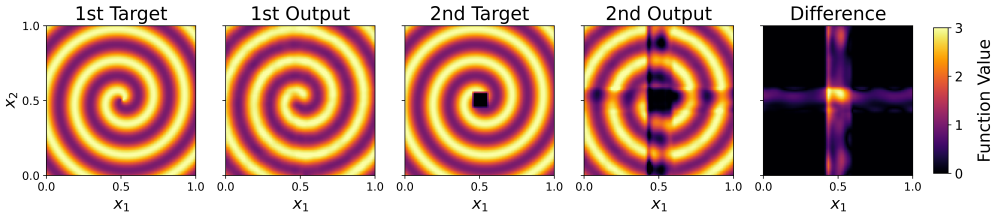


Figure 6: Visual inspection of target functions and model outputs over Task 1 and Task 2.

8 Conclusion

The main contribution of the paper is theoretical and technical. A representation theorem is presented that outlines how to approximate multi-variable functions with single-variable functions, and exponential functions. Atlas approximates all arbitrary single-variable functions with mixed-density B-spline functions. Atlas is constructed in such a way that the gradient vector with respect to trainable parameters is bounded, regardless of how large an Atlas model is. Atlas was shown to exhibit intrinsic memory retention, except for a small cross-shaped region of interference and off-target effects.

The chosen experiment demonstrated the expansion abilities of Atlas, and its memory retention which is interesting. The specific target function is well-studied. It was mainly chosen for demonstration purposes. Higher-dimensional tests are needed to evaluate practical utility of Atlas. Since many of the basis functions are zero, the presented implementation of Atlas does redundant calculations. Improving efficiency is an ongoing research goal.

As far as societal impacts are concerned: It is possible that Atlas could allow for the creation of more powerful machine learning algorithms, that require less resources to train and deploy. Further testing is needed to make any concrete claim.

References

[1] K. Branson. A practical review of uniform b-splines, 2004.

[2] A. S. Douzette. B-splines in machine learning. Master's thesis, Department of Mathematics, University of Oslo, 2017.

[3] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[4] R. Hadsell, D. Rao, A. A. Rusu, and R. Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences*, 24(12):1028–1040, 2020.

[5] B. Hanin. Which neural net architectures give rise to exploding and vanishing gradients? *Advances in neural information processing systems*, 31, 2018.

[6] B. Hanin. Universal function approximation by deep neural nets with bounded width and relu activations. *Mathematics*, 7(10):992, 2019.

[7] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[8] P. Kaushik, A. Gain, A. Kortylewski, and A. Yuille. Understanding catastrophic forgetting and remembering in continual learning with optimal relevance mapping. *CoRR*, abs/2102.11343, 2021. URL `https://arxiv.org/abs/2102.11343`.

[9] R. Kemker, M. McClure, A. Abitino, T. Hayes, and C. Kanan. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[10] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. 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`.

[11] A. Kratsios. The universal approximation property: Characterization, construction, representation, and existence. *Annals of Mathematics and Artificial Intelligence*, 89(5):435–469, 2021. doi: 10.1007/s10472-020-09723-1.

[12] S. H. Lane, M. Flax, D. Handelman, and J. Gelfand. Multi-layer perceptrons with b-spline receptive field functions. In *Advances in Neural Information Processing Systems*, pages 684–692, 1991.

[13] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[14] K. McRae and P. A. Hetherington. Catastrophic interference is eliminated in pretrained networks. In *Proceedings of the 15h Annual Conference of the Cognitive Science Society*, pages 723–728, 1993.

[15] A. Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2): 123–146, 1995.

[16] S. Scardapane, M. Scarpiniti, D. Comminiello, and A. Uncini. Learning activation functions from data using cubic spline interpolation. In *Italian Workshop on Neural Nets*, pages 73–83. Springer, 2017.

[17] Y. Shin and J. Ghosh. The pi-sigma network: an efficient higher-order neural network for pattern classification and function approximation. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, volume i, pages 13–18 vol.1, 1991. doi: 10.1109/IJCNN.1991.155142.

[18] H. van Deventer, P. J. van Rensburg, and A. Bosman. Kasam: Spline additive models for function approximation, 2022. URL `https://arxiv.org/abs/2205.06376`.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? **[TODO]**
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? **[TODO]**
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? **[TODO]**
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? **[TODO]**
   (b) Did you mention the license of the assets? **[TODO]**
   (c) Did you include any new assets either in the supplemental material or as a URL? **[TODO]**
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]