MA 591 - HOMEWORK 04

Hayden Outlaw

NCSU Applied Mathematics, 7 Nov 2025

---

1 δ-Net and 0-th Order Optimization

Let $f : [0,1]^n \to \mathbb{R}$ be an L-Lipschitz continuous function satisfying:

$$|f(x) - f(y)| \leq \|x - y\|_2, \;\; \forall\, x, y \in [0,1]^n$$

Suppose that we are interested in solving:

$$\min_{x \in [0,1]^n} f(x)$$

and $x^* \in [0,1]^n$ is a global minimizer. Suppose we can only evaluate the function values on $[0,1]^n$, so we consider a set of gridded collocation points:

$$\mathcal{G} = \left\{ z := \left(\frac{i_1}{N}, \frac{i_2}{N}, \ldots, \frac{i_m}{N} \right)^\top \in [0,1]^n : i_1, i_2, \ldots, i_n \in \{0, 1, \ldots, N\} \right\}$$

1.1 Establishing δ-Net

We can show $\mathcal{G}$ is a δ-net with $\delta = \frac{\sqrt{n}}{2N}$. First, for any $x \in [0,1]^n$, with entries x_i, we need to consider the closest $z \in \mathcal{G}$. Since the entries of $\mathcal{G}$ are arranged on a regular grid, we can first establish that for each entry x_i, that there exist i_k, i_{k+1} such that:

$$\frac{i_k}{N} \leq x_i \leq \frac{i_{k+1}}{N}$$

With this, for any entry x_i, the maximum distance to the nearest grid coordinate can be established as:

$$|x_i - z_i| \leq \frac{1}{2N}$$

Now, to establish the vector distance bound, we can look at the entries of $x - z \in \mathbb{R}^n$:

$$\begin{aligned} \|x - z\|_2^2 &= \sum_{i=1}^{n} |x_i - z_i|^2 \\ &\leq \sum_{i=1}^{n} \left(\frac{1}{2N} \right)^2 \\ &= n \left(\frac{1}{2N} \right)^2 \end{aligned}$$

Since these are positive values, we can take square roots and conclude:

$$\|x - z\|_2 \leq \frac{\sqrt{n}}{2N} = \delta$$

1.2 Establishing Optimization Error

Let z^* be the solution to $\min_{z\in\mathcal{G}} f(x)$. Let M bet he number of points in $\mathcal{G}$. We can show that:

$$f(z^*) \leq f(x^*) + L\frac{\sqrt{n}}{2(M^{\frac{1}{n}}-1)}$$

First, we need to relate the fidelity of the δ-Net to the number of sample points. If the grid points are spaced every $\frac{1}{N}$, there are N^n+1 total points[1], which we define to be M. So, taking the n-th root of both sides, we have that $N = M^{\frac{1}{n}} - 1$. Now, using the Lipschitz continuity of f:

$$\begin{aligned} f(z^*) - f(x^*) &\leq L\|z^* - x^*\|_2 \\ &\leq L\frac{\sqrt{n}}{2N} \text{ (via 1.1)} \\ &= L\frac{\sqrt{n}}{2(M^{\frac{1}{n}}-1)} \end{aligned}$$

1.3 Relating Sample Count to Error

If $n = 100$, $L = 2$, how many grid points M should be generated in order to guarantee $f(z^*)$ is close to $f(x^*)$ within the error 10^{-3}?

Using the result from 1.2, if we know $f(z^*) - f(x^*) \leq \frac{L\sqrt{n}}{2(M^{\frac{1}{n}}-1)}$, then:

$$\begin{aligned} \frac{2\sqrt{100}}{2(M^{\frac{1}{2}}-1)} \leq 10^{-3} &\implies \\ 20 \leq 10^{-3}(2M^{\frac{1}{2}} - 2) &\implies \\ 20000 \leq 2M^{\frac{1}{2}} - 2 &\implies \\ 10000 - 1 \leq M^1 2 &\implies \\ (10000-1)^2 \leq M & \end{aligned}$$

So, the number of grid points should be greater than or equal to $(10000-1)^2$ to guarantee a collocation error less than or equal to 10^{-3}.

2 Partitions of Unity

2.1 Continuous Exponential Bases

Consider the following function:

$$f(x) = \begin{cases} Ce^{\frac{-1}{1-\|x-x^*\|_2^2}}, & \text{if } x \in B_1(x^*) := \{z \in \mathbb{R}^n : \|z - x^*\|_2 < 1\} \\ 0 \text{ otherwise} \end{cases}$$

where C is chosen such that $f(x^*) = 1$. We can show that f is continuous on $\mathbb{R}^n$. We use the 2 norm, but since $\mathbb{R}^n$ is finite dimension all norms are equivalent and equally valid for analysis.

We divide into three cases. If x, y $B_1(x^*)$, the conclusion follows immediately from the continuity of the 2-norm, and if both $x, y \notin B_1(x^*)$, then $f(x) = f(y) = 0$, and the conclusion is trivial. So, we know f is continuous on the open set $B_1(x^*)$, and $\mathbb{R}^n \setminus B_1(x^*)$. However, since $B_1(x^*)$ is an open set, we must still prove f is closed on $\text{Cl}[B_1(x^*)]$, or on the closure of the ball around x^*.

Since $B_1(x^*)$ is a ball, we know the closure $\overline{B_1(x^*)} = \{z \in \mathbb{R}^n : \|z - x^*\|_2 \leq 1\}$. We already know f is continuous on the interior of $B_1(x^*)$, so we only need to check the boundary: $\partial B_1(x^*) = \{z \in \mathbb{R}^n : \|z - x^*\| = 1\}$. We know for all $z \in \partial B_1(x^*)$, $f(z) = 0$, which is also equal to $f(z)$ for all $z \notin B_1(x^*)$, so f is clearly continuous to the boundary from the outside of the ball. Finally, to show f is continuous to the boundary from the inside of the ball, if x is in the interior of the ball, and y is on the surface, by the triangle inequality, we have that:

$$\|x^* - y\|_2 \leq \|x^* - x\|_2 + \|x - y\|_2 \implies 1 - \|x^* - x\|_2 \leq \|x - y\|_2$$

[1] accounting for the coordinate at the origin

Since f is continuous on the interior, for any limit in the interior of the ball $\lim x_n \to y$, we have $\|x_n - y\|_2 \to 0$ implies $1 - \|x^* - x\|_2 \to 0$, and therefore:

$$\lim_{x_n \to y} \|f(x_n) - 0\| = 0$$

as the negative exponential will go to 0, and C is fixed. Therefore, since $\lim_{x_n \to y} \|f(x_n) - f(y)\| = 0$, the function is also continuous to the boundary from the inside of the ball. The open ball $B_1(x^*)$, its boundary, and its exclusion partition $\mathbb{R}^n$, so we've accounted for all cases, and can conclude f is continuous on all of $\mathbb{R}^n$.

2.2 Interpolation via Exponential Bases

We can use f as defined in 2.1 to define a basis function for interpolation. For N points $\{x_i\}_{i=1}^n$, and a given minimum radius δ, define $r = \min\{\frac{\|x_i - x_j\|_2}{2}, \delta\}$, ie the minimum distance between any two points in 2 norm, and the radius. Then:

$$f_i(x) = \begin{cases} C_i e^{\frac{-r^2}{r^2 - \|x - x_i\|_2^2}}, & \text{if } x \in B_1(x_i) := \{z \in \mathbb{R}^n : \|z - x_i\|_2 < r\} \\ 0 \text{ otherwise} \end{cases}$$

which is similarly normalized such that $f_i(x_i) = 1$. Since these are Gaussian functions, that restricts $0 \leq f_i(x) \leq 1$ for all i. The reason we are setting the 'radius' to r is so that f_i interpolate $\{x_i\}$ such that $f_j(x_i) = \delta_{ij}$ Now we can take these radial basis functions, and normalize them such that:

$$\phi_i(x) = \frac{f_i(x)}{\sum_{j=1}^n f_j(x)}$$

We then have:

- $\sum_{j=1}^n \phi_i(x) = 1$ by construction
- $\phi(x_i) = \frac{f(x_i)}{f(x_i)} = 1$ since f_i interpolate $\{x_i\}$
- $0 \leq \phi_j(x) \leq 1 \; \forall j$, as f_j is normalized such that $0 \leq f_j(x) \leq 1$
- $\begin{cases} \phi_j(x) > 0 \text{ if } x \in B_\delta(x_j) \cap [0,1]^n \\ \phi_j(x) = 0 \text{ otherwise} \end{cases}$, as we set the radius limit to be less than or equal to δ, for any δ-covering of $[0,1]^n$ around the data points.

2.3 Uniform Convergence of Interpolation Series

Let $u \in C[0,1]^n$. Using the results from 2.1, 2.2, we can show that for any $\varepsilon > 0$, there exist N continuous functions $\{\phi_j\}_{j=1}^n$ such that

$$\sup_{x \in [0,1]^n} |u(x) - u_N(x)| < \varepsilon$$

where $u_N(x) = \sum_{j=1}^N u(x_j)\phi_j(x)$.

$[0,1]^n$ is closed and bounded, so by the Heine-Borel theorem, it is a compact subset of $\mathbb{R}^n$. Therefore, for any $\delta > 0$, the set has some finite cover such that there are finitely many $\{x_i\}_{i=1}^N$ such that:

$$[0,1]^n \subset \bigcup_{i=1}^N B_\delta(x_i)$$

If we define the bases ϕ_i around these points, as defined in 2.2, with radius δ, since we know any x must be in at least one $B_\delta(x_i)$, there is always some x_i such that $|x - x_i| < \delta$. Now, let x_i be the closest radial center to x, and y_i be the closest radial center to y, for any $x, y \in [0,1]^d$:

$$\begin{aligned} |u_N(x) - u_N(y)| &= |u_N(x) - u_N(x_i) + u_N(x_i) - u_N(y) + u_N(y_i) - u_N(y_i)| \\ &\leq |u_N(x) - u_N(x_i)| + |u_N(x_i) - u_N(y_i)| + |u_N(y) - u_N(y_i)| \\ &= |u_N(x) - u_N(x_i)| + |u(x_i) - u(y_i)| + |u_N(y) - u_N(y_i)| \;\; (\text{as } u_N \text{ interpolates data}) \end{aligned}$$

Since $|x - x_i| < \delta$, $|y - y_i| < \delta$, and we know $u \in C[0,1]^n$, by setting each of these terms to $\frac{\varepsilon}{3}$, we know there is some δ-cover such that each of these three terms is less than or equal to $\frac{\varepsilon}{3}$, and so we have that $\{u_N(x)\}$ is an equicontinuous sequence on $[0,1]^d$.

Since we know $\{u_N\}$ is a set of continuous functions on a compact set, this is also a uniformly bounded sequence. Therefore, by the Arzela-Ascoli theorem, $\{u_N\}$ converges to its limit uniformly as $N \to \infty$.

Finally, since $[0,1]^n$ is compact, the limit as $N \to \infty$ is taking essentially an infinite number of data points - ie we have u_N perfectly interpolating every x_i in the domain, so we have that $\lim_{N\to\infty} u_N(x) = u(x)$ on $[0,1]^n$.

3 DeepONet Implementation

Consider the 1D parametric inviscid Burgers' equation:

$$\begin{cases} u_t + uu_x = 0, \ (x,t) \in (-3,3) \times (0,2], \\ u(-3,t) = u(3,t) \ \forall \ t \in (0,2], \\ u(x,0) = \alpha e^{\frac{-x^2}{2\omega^2}}, \ \forall \ x \in (-3,3) \end{cases}$$

Where $\mu = (\alpha, \omega) \in [0.7, 0.8] \times [0.9, 1.0]$ represents the parameters of interest. The operator we want to learn is:

$$\mathcal{G} : \mu \to u$$

where u is the solution to the corresponding Burgers' equation. We use synthetic pre-generated data[2] that consists of a $201^2 \times 2$ equidistant collocation grid on the domain, as well as samples of μ with the corresponding solutions $u(x,t;\mu)$ on that dataset.

We will use a Deep Operator Network (DeepONet) to learn an approximation to $\mathcal{G}$. For the model, across all experiments these parameters are fixed:

1. Total DeepONet Width: 16
2. DeepONet Trunk Architecture: Depth 4, Width 100, and activation `gelu`
3. DeepONet Branch Architecture: Depth 4, Width 100, and activation `tanh`
4. Unless otherwise specified, the `adam`[3] optimizer with a 10^{-3} learning rate across $20,000$ training iterations to solve the optimization problem.

We then use the relative L_2 error to evaluate the quality of the solutions across the entire (x,t) grid in the specified domain.

Computation was performed on an ARM M1 Pro chip using the JAX software package, with Apple Metal compilation. Implementation details and source codes are available at: *https://github.com/outlawhayden/ncsu-sciml/tree/main/hw4*

[2] on course page
[3] as implemented in Optax

3.1 Unmodified DeepONet

First, using an unmodified DeepONet, we achieve a minimum relative L^2 loss of $1.4512 * 10^{-4}$, with a total training time of 5.2 minutes. Depicted in Figure 1 is the training trajectory of the model with respect to the unmodified training loss. While there are the loss spikes characteristic of first-order optimization algorithms such as `adam` on highly nonconvex loss surfaces, the minimum loss value does decay down to a scale of 10^{-4}, which is within visual inspection tolerance. Figure 3 depicts the first four learned basis functions - ie the first four output columns of the 'trunk' model given the input of the entire discretized domain. In this case, the input evaluation to generate these results is the same as the training collocation points, since they are both equidistant grids of fine resolution. On visual inspection they seem fairly smooth, and alternate to capture the dynamics of the propagating shockwave as t increases, but beyond that there is no clear relationship between them.

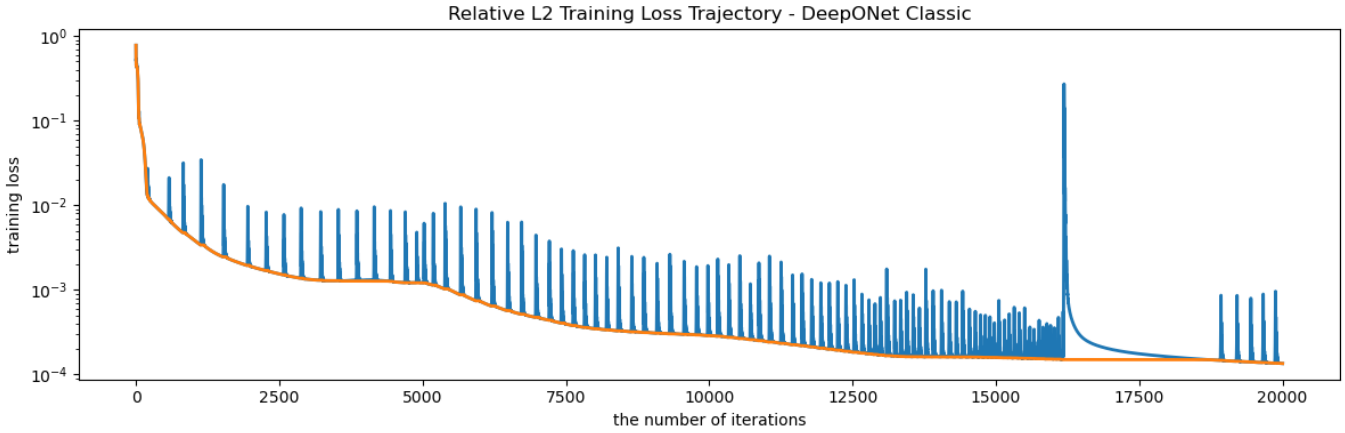

Figure 1: Training Loss - DeepONet

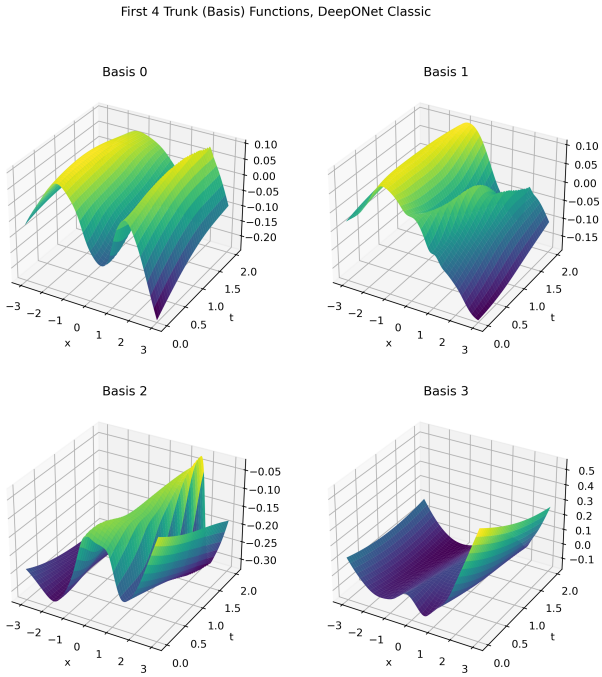

Figure 2: First Four Learned Bases - DeepONet

Perhaps more illustrative is the performance in Figure 3, which is the predicted solution u_{NN} on the discretized domain, compared to the provided ground truth labels. As with most numerical solutions to Burger's equation, most of the error is concentrated in the area near where the shockwave propagation becomes increasingly stiff at large time values. Especially since the training data is restricted to an equidistant grid, the model suffers the most where multiscale resolution is the most valuable in terms of accuracy at a given sample point. The resulting modified networks are an attempt to mitigate this error around increasingly stiff areas in the domain, and provide higher quality multiscale predictions.

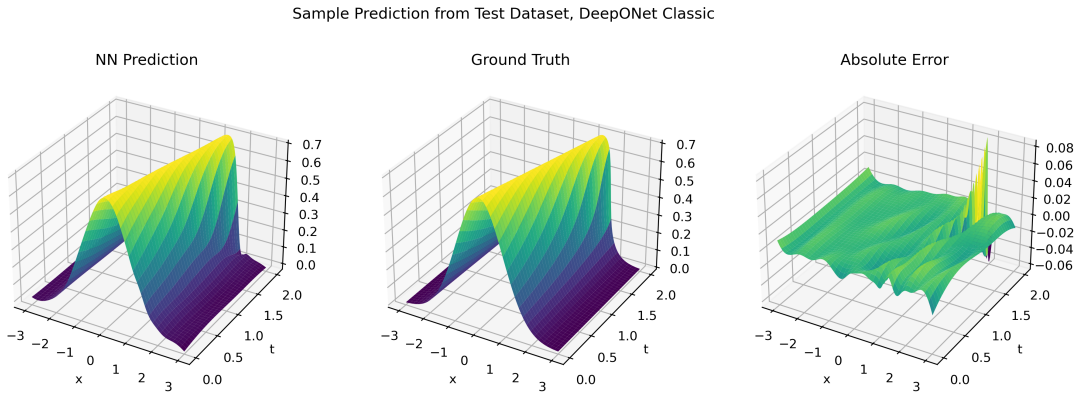

Figure 3: Sample Prediction - DeepONet

3.2 DeepONet with Residual Based Attention

Next, with the addition of residual based attention in the loss function, we obtain a total minimum relative L^2 error of $7.3606 * 10^{-6}$, with a total training time of 5.31 minutes. In using the residual based attention mechanism in the training loss, we initialize the attention vector $\lambda = \mathbf{0}$, or as the vector of all 0. We also use the same learning rate of 10^{-3}, and parameter $\gamma = 0.999$.

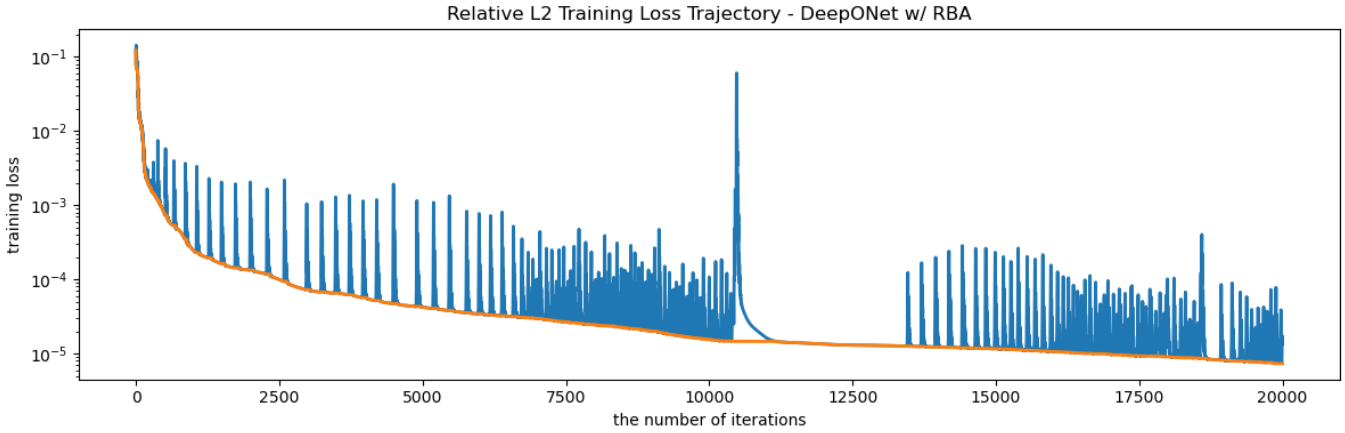

Figure 4: Training Loss - DeepONet w/ RBA

The training trajectory with the relative L^2 error is depicted in Figure 4 - compared to the classical DeepONet architecture, the modified loss clearly presents as a noisier training process, with a similar final loss value measured in the entire domain. The learned bases in Figure 5 are also of similar interpretability, but the largest difference in performance comes in the quality of literal solution predictions. As shown in Figure 6, even if the solution is marginally less accurate on more stable parts of the domain, the loss spike around the shockwave areas is reduced where the solution is nearly singular.

The weighting of the loss proportional to model accuracy means that the loss function is most notably valued more at the area where there are known multiscale resolution issues. While the overall loss metric did decrease by an order of magnitude, that average measurement obfuscates the prediction quality at different parts of the domain. Given that this experiment used the same training time and parameters as the classical model, the uneven RBA weighting could also potentially explain the deterioration of the solution in other parts of the domain, as those areas are now weighted less. But as evidenced by the slight downward training loss trajectory, these slight inaccuracies located far from stiff areas of the solution could be resolved with further training time.

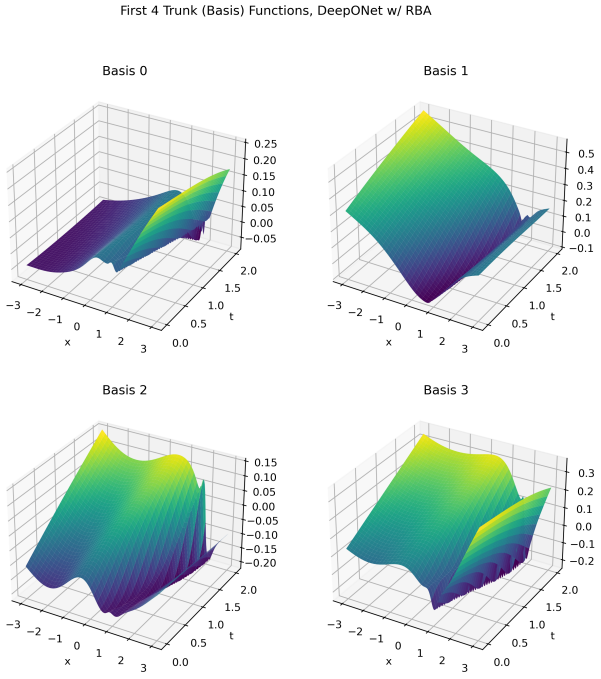

Figure 5: First Four Learned Bases - DeepONet w/ RBA

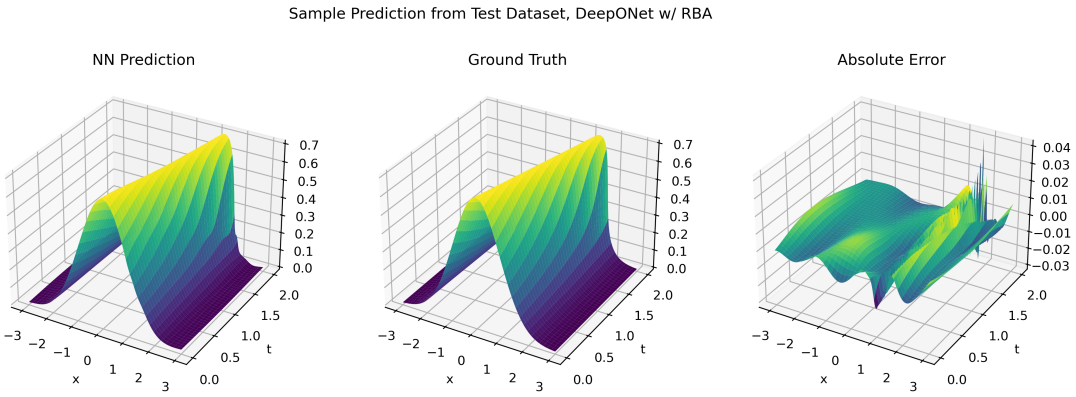

Figure 6: Sample Prediction - DeepONet w/ RBA

3.3 DeepONet with 2 Step Training

Next, if we utilize the two-step training protocol with an unmodified DeepONet, and orthogonalize the corresponding basis functions between steps, we achieve a total relative L^2 error of $3.381 * 10^{-6}$, with a total training time of 2.13 minutes. Compared to the original training structure, as shown in Figure 7, this results in a much more regular and predictable training pattern for the trunk network, with smaller spikes in the training loss across iterations.

In this configuration, the trunk network was trained across $20,000$ iterations, and the branch network over $150,000$ iterations - since the components no longer need to have the same training schedule. However, both still utilize the `adam` optimization algorithm with the same learning rate of 10^{-3}. More notable in this setting is the change in the learned basis functions in Figure 9 - while they are less regular, they are orthogonalized. This is difficult to intuit visually in a higher dimensional space, but they form a better conditioned basis for the solution space, and their change in spatial frequency at higher times to encompass the propagating shockwave is more visible than in the other cases.

Compared to the addition of RBA to the original model, the two-step training regime results in a similar overall relative L^2 loss, on the order of 10^{-6}, as well as similar sample predictions as shown in Figure 10. This is a roughly equivalent mitigation of the multiscale resolution error induced by the propagating shockwave - but note that these are philosophically very different

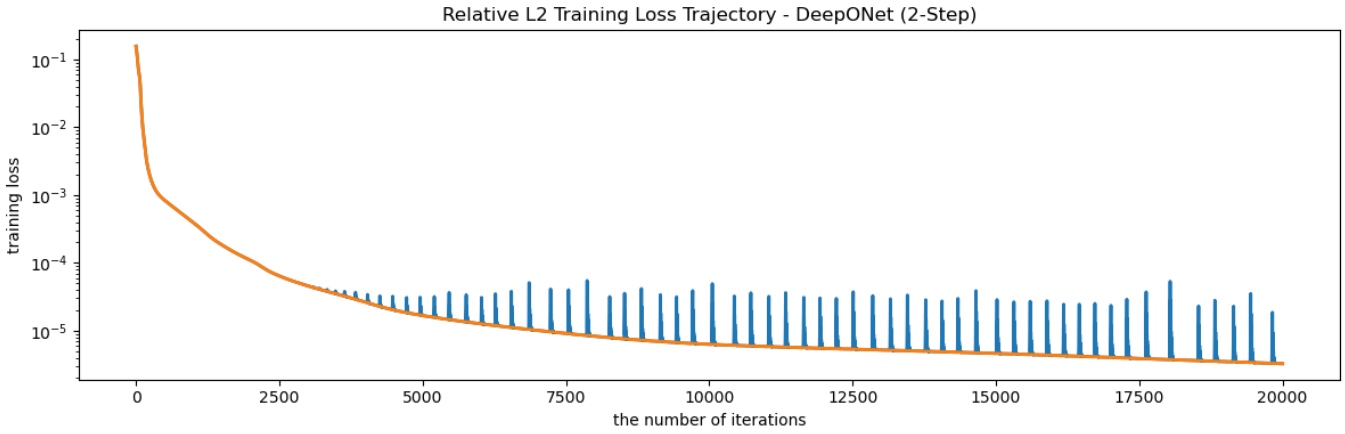

Figure 7: Training Loss - DeepONet (2-Step)

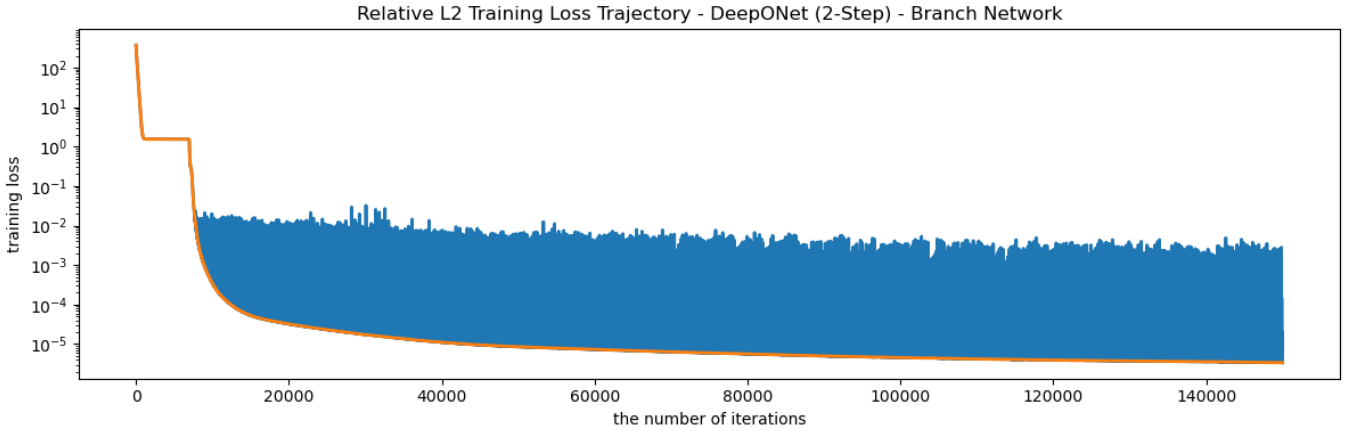

Figure 8: Training Loss - DeepONet (2-Step) - Branch Network

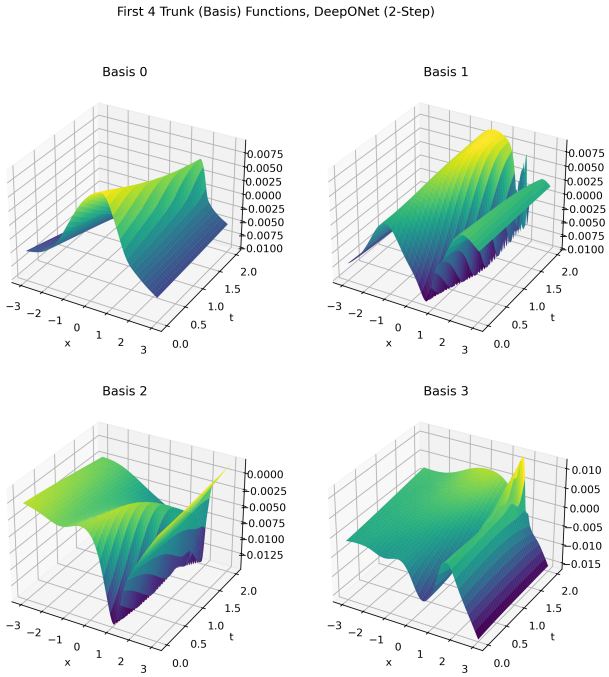

Figure 9: First Four Learned Bases - DeepONet (2-Step)

mechanisms of addressing multiscale resolution. This utilizes the same loss function, and doesn't require adding hyperparameters to tune, although it comes at the cost of increased training time and implementation complexity.

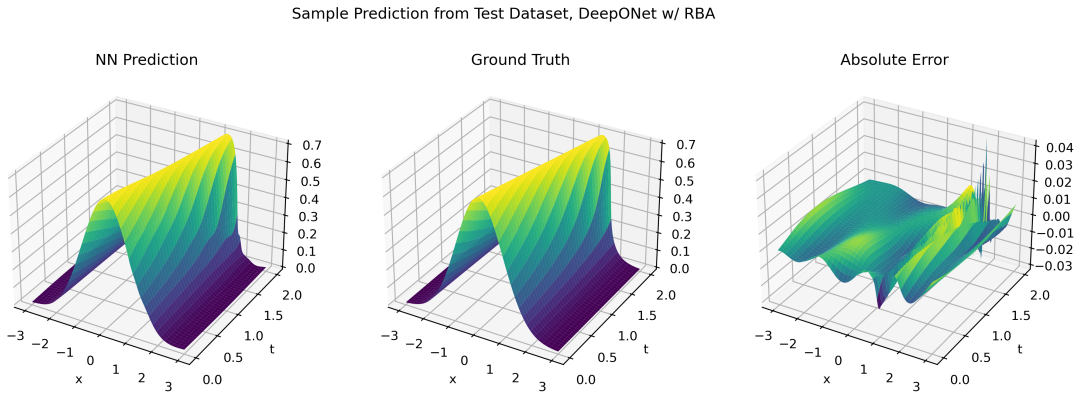

Figure 10: Sample Prediction - DeepONet (2-Step)

3.4 DeepONet with Residual based Attention and 2 Step Training

Finally, if we include residual based attention in the training loss, alongside the two-step training regimen, we obtain a total relative L^2 loss of $1.21 * 10^{-5}$, with a total training time of 2.21 minutes. Although the overall metric loss on the domain is slightly higher, the goal of this experiment is to see if the two mechanisms for adressing multiscale inaccuracies, orthognalization in training and residual based attention, are compatible with one another. The training loss across iterations, as shown in Figure 11, seems to be more stable than the case only utilizing RBA, but more sensitive than the case only utilizing the two-step training regimen.

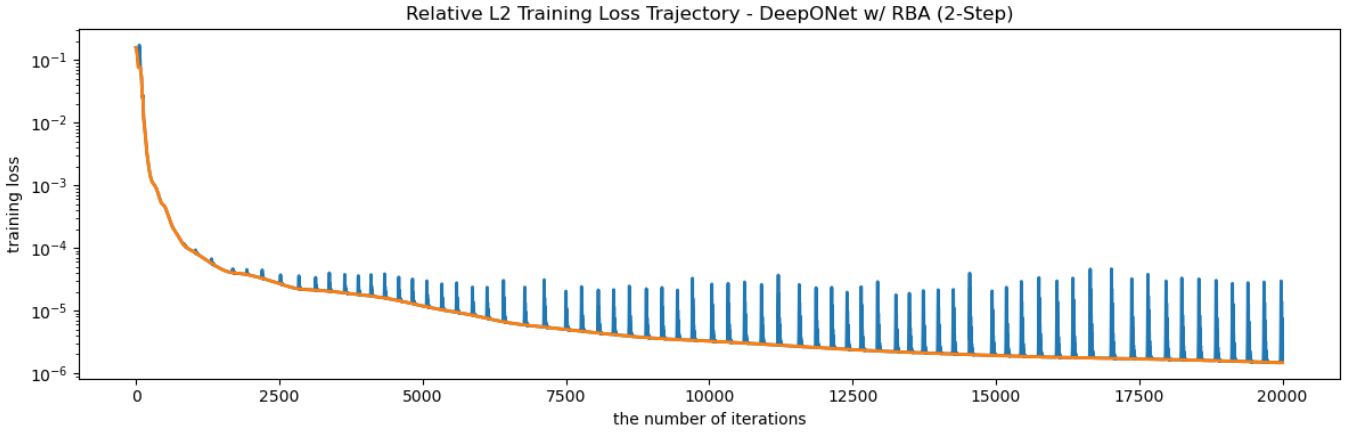

Figure 11: Training Loss - DeepONet w/ RBA(2-Step)

The learned basis functions, as depicted in Figure 13[4], are still orthogonal, but slightly more irregular. Most critically though, as shown on a sample prediction in Figure 14, the error around the shockwave propagation is drastically reduced. While error on the rest of the domain perhaps increases slightly, the two strategies together form an effective mechanism for mitigating the increasingly stiff components of the solution, which is classically the most difficult part of generating a numerical solution for this particular system.

[4]Note the triangulation artifacts are from the $3D$ render from the interpolation, not an attribute of the functions themselves.

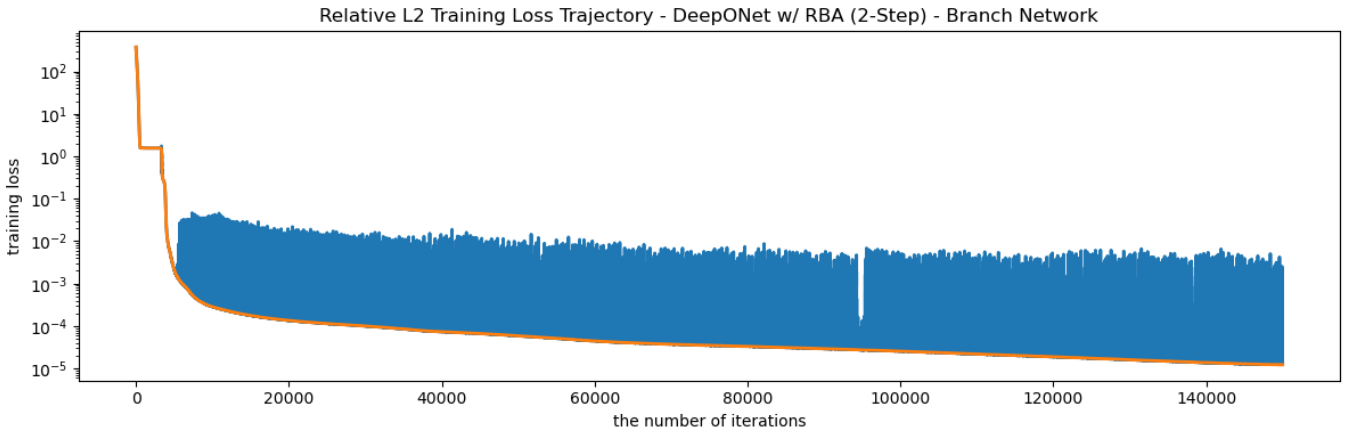

Figure 12: Training Loss - DeepONet w/ RBA(2-Step) - Branch Network

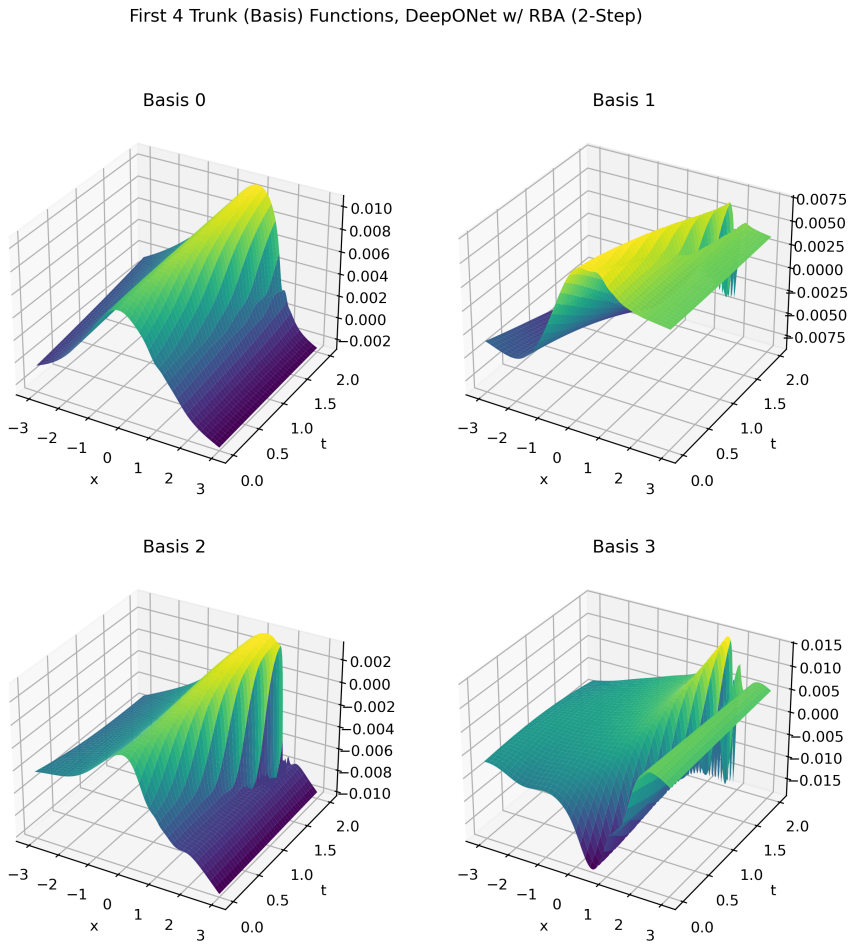

Figure 13: First Four Learned Bases - DeepONet w/ RBA (2-Step)

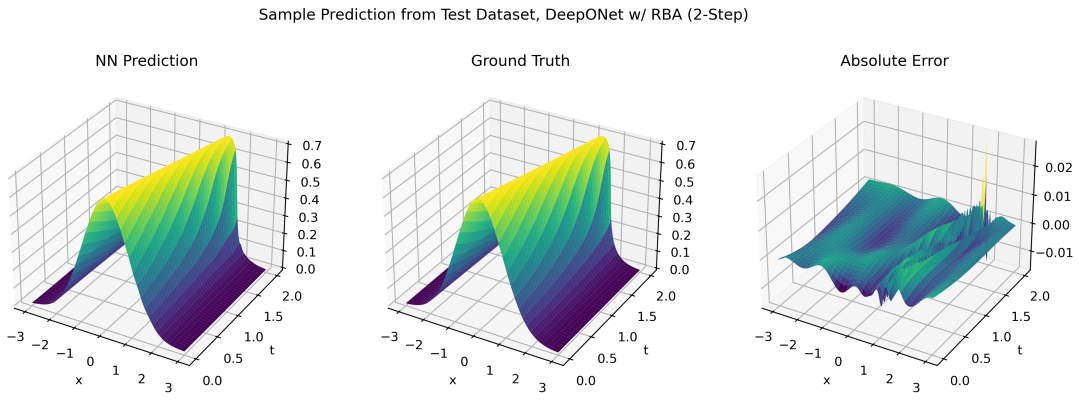

Figure 14: Sample Prediction - DeepONet w/ RBA (2-Step)