Proceedings of Machine Learning Research vol XX:1–21, 2023

Operator Learning for Nonlinear Adaptive Control: Supplemental Materials

Luke Bhan LBHAN@UCSD.EDU
Department of Electrical and Computer Engineering, University of California, San Diego

Yuanyuan Shi YYSHI@ENG.UCSD.EDU
Department of Electrical and Computer Engineering, University of California, San Diego

Miroslav Krstic MIKRSTIC@UCSD.EDU
Department of Mechanical and Aerospace Engineering, University of California, San Diego

1. Experimental System I: Linear Scalar Plant

1.1. Problem Statement

Consider the simple toy example of a linear scalar plant

$$\dot{y} = \theta y \tag{1}$$

where y(t) is the measured state and the parameter θ is unknown. Note this toy system is analytically solvable into an exponential of the form $y(t) = y_0 e^{\theta t}$ We derive a parameter estimator for the scalar system following Ioannou and Sun (1995).

$$\dot{\alpha} = -y^2(y^2 + \alpha) \tag{2}$$

$$\hat{\theta} = \frac{1}{2}(y^2 + \alpha) \tag{3}$$

where $y(t)$ is the measurement, $\alpha(y, t)$ is an internal state that depends on an initial condition α_0 and $\hat{\theta}(y, t)$ is the estimate of the true value of θ. The full derivation and analysis is available in Section 1.2. Additionally, using the variation of constants formula, we can obtain a direct input-output formula for the estimator in Eqn. 2, 3 as:

$$\hat{\theta}(t) = \frac{1}{2}\left[\alpha_0 e^{-\int_0^t y^2(\tau)d\tau} + y^2(t) - \int_0^t e^{-\int_\tau^t y^2(\sigma)d\sigma} y^2(\tau)d\tau\right] \tag{4}$$

We consider a variety of different solutions of Eqn. 1, 4 and supply an example solution and estimate in Figure 1.1 where $\theta > 0$ to satisfy the persistent excitation condition found in Section 1.2.

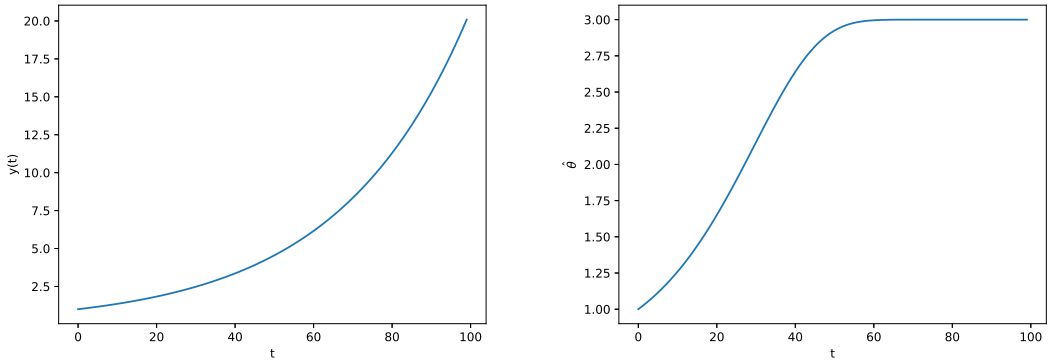

Figure 1: An example solution of the scalar plant system in Eq 1 and the parameter estimation in Eq 4 where $\theta = 3$, $y_0 = 1$, $\alpha_0 = 1$

1.2. Estimator Derivations and Analysis

1.2.1. DERIVATION OF ESTIMATOR

We begin by deriving the estimator in Eqn. 2, 3 according to Ioannou and Sun (1995). Recall the system from Eqn. 1:

$$\dot{y} = \theta y \tag{5}$$

We define our loss as the following minimization problem:

$$\min_{\hat{\theta}} L(\hat{\theta}) = \frac{1}{2}(\hat{\theta}y - \dot{y})^2 \tag{6}$$

Taking the gradient as:

$$\nabla_{\hat{\theta}} L(\hat{\theta}) = y(\hat{\theta}y - \dot{y}) \tag{7}$$

Applying gradient flow:

$$\dot{\hat{\theta}} = -y(\hat{\theta}y - \dot{y}) \tag{8}$$

Now, we do not have the measurement for $\dot{y}(t)$, so we remove it through the following:

$$\dot{\hat{\theta}} - y\dot{y} = -\hat{\theta}y^2 \tag{9}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}(\hat{\theta} - \frac{1}{2}y^2) = -\hat{\theta}y^2 \tag{10}$$

Now, let $\alpha = 2(\hat{\theta} - \frac{1}{2}y^2)$. Then we have:

$$\hat{\theta} = \frac{1}{2}(y^2 + \alpha) \tag{11}$$

$$\dot{\alpha} = 2\dot{\hat{\theta}} - 2y\dot{y} \tag{12}$$

Substituting for $\dot{\hat{\theta}}$:

$$\dot{\alpha} = -2\hat{\theta}y^2 + 2y\dot{y} - 2y\dot{y} = -2\hat{\theta}y^2 = -y^2(y^2 + \alpha) \tag{13}$$

Eqn. 11 and 13 represent the estimator in Section 1.1 above.

1.2.2. ANALYSIS OF ESTIMATOR

We aim to analyze this estimator and derive the conditions for persistent excitation to ensure we chose θ as well posed for the estimator (Ie θ generates enough signal for us to properly estimate).

Denote the following:

$$\tilde{\theta} = \theta - \hat{\theta} \tag{14}$$

Taking the derivative:

$$\dot{\tilde{\theta}} = 0 - \dot{\hat{\theta}} \tag{15}$$

And substituting for $\dot{\hat{\theta}}$ using Eqn 8

$$\dot{\tilde{\theta}} = -(y\dot{y} - \hat{\theta}y^2) \tag{16}$$

$$\dot{\tilde{\theta}} = -(y\theta y - \hat{\theta}y^2) \tag{17}$$

$$\dot{\tilde{\theta}} = -(\theta y^2 - \hat{\theta}y^2) \tag{18}$$

$$\dot{\tilde{\theta}} = -(y^2\tilde{\theta}) \tag{19}$$

Define a valid Lypunov function as the following:

$$V = \frac{1}{2}\tilde{\theta}^2 \tag{20}$$

Taking the derivative, we obtain:

$$\dot{V} = \tilde{\theta}\dot{\tilde{\theta}} \tag{21}$$

Substituting Eqn. 19 from above:

$$\dot{V} = -(y^2\tilde{\theta})\tilde{\theta} \tag{22}$$

$$\dot{V} = -y^2\tilde{\theta}^2 \leq 0 \tag{23}$$

Since, we do not have that the derivative is strictly less than 0, we cannot conclude asymptotic stability from Lypunov's theorem. Instead, we will show that the function V converges to a constant and then present the conditions for when that constant will lead to convergent solutions for θ. We know that the function $V \geq 0$ is a non-increasing function of time. Thus, we know that $\lim_{t\to\infty} V(\tilde{\theta}(t)) = V_\infty$ exists. Substituting for V yields the following expression:

$$\lim_{t\to\infty} \tilde{\theta}(t)^2 = 2V_\infty \tag{24}$$

which implies that $\hat{\theta}$ converges with time. (Since we have θ as constant and $\tilde{\theta} = \theta - \hat{\theta}$). However, it is not guaranteed that $\hat{\theta}$ converges to θ without extra conditions. To find these conditions, we solve Eqn. 19 for a closed form of $\tilde{\theta}$.

$$\tilde{\theta}(t) = e^{-\int_0^t y^2(\tau)d\tau}\tilde{\theta}(0) \tag{25}$$

From, here we know the closed form of y is $y_0e^{\theta t}$ leading to

$$\tilde{\theta}(t) = e^{-\int_0^t y_0e^{2\theta\tau}d\tau}\tilde{\theta}(0) \tag{26}$$

To force Eqn. 26 to 0, we need the exponent in the first term to have a large negative value and thus we will only obtain this if $\theta > 0$. In this case, our estimator for $\hat{\theta}$ will tend toward θ asymptotically and we denote the condition $\theta > 0$ as a requirement for our training data.

1.3. Data Generation and Hyperparameters

To build a dataset, we generate 1000 solutions for $\theta \sim \text{uniform}[0, 5]$, $\alpha_0 \sim \text{uniform}[0, 10]$, and $y_0 \sim \text{uniform}[0, 10]$. Additionally, we consider solutions from time $t \in [0, 1]$ where $\Delta t = 0.01$ and $nt = 100$. We then split the 1000 instances into 900 solutions for training and 100 solutions for testing and attempt to learn the mapping from:

$$[\alpha_0, y(0), y(\Delta t), y(2\Delta t), ..., y(t = 1)] \mapsto [\hat{\theta}(0), \hat{\theta}(\Delta t), \hat{\theta}(2\Delta t), ..., \hat{\theta}(t = 1)] \tag{27}$$

We consider 4 various architectures for training our models. We present their hyper-parameters in Table 1.

Model	Network Architecture	Optimizer	Number of Parameters
Gated Recurrent Unit (GRU) Cho et al. (2014)	GRU(hiddenSize=400) $\mapsto$ Linear(width=1)	Adam(LR=0.01, $\beta = (0.9, 0.999)$, WeightDecay=0.0001) LRScheduler.stepLR(γ=0.5, stepSize=50)	485201
Long Short-Term Memory (LSTM) Hochreiter and Schmidhuber (1997)	LSTM(hiddenSize=400) mapsto Linear(width=1)	Adam(LR=0.01, $\beta = (0.9, 0.999)$, WeightDecay=0.0001) LRScheduler.stepLR(γ=0.5, stepSize=50)	646801
DeepONet Lu et al. (2021)	Branch: GRU(hiddenSize=256) Trunk: Linear(width=64) $\mapsto$ ReLU $\mapsto$ Linear(width=128) $\mapsto$ ReLU $\mapsto$ Linear(width=256) $\mapsto$ ReLU	Adam(LR=0.01, $\beta = (0.9, 0.999)$, WeightDecay=0.0001) LRScheduler.stepLR(γ=0.5, stepSize=50)	98528
Fourier Neural Operator (FNO) Li et al. (2020)	Linear (width=32) $\mapsto$ FFT (12 Modes) $\mapsto$ Linear (width=32) $\mapsto$ GeLU $\mapsto$ Linear (width=32) $\mapsto$ GeLU $\mapsto$ Linear (width=32) $\mapsto$ GeLU $\mapsto$ IFFT (12 Modes) $\mapsto$ Linear (width=1)	Adam(LR=0.01, $\beta = (0.9, 0.999)$, WeightDecay=0.0001) LRScheduler.stepLR(γ=0.5, stepSize=50)	107009

Table 1: Hyperparameters for various network architectures for the simple scalar plant. FFT and IFFT denote Fast Fourier Transform and Inverse Fast Fourier Transform respectively. LR denotes the Learning Rate.

1.4. Results

We train each architecture on the same training and testing data using the aforementioned 900/100 split for 1000 epochs. We begin by showing the loss functions from all 4 network architectures in Figure 1.4. It is clear that every architecture converges by the end of the 1000 epochs.

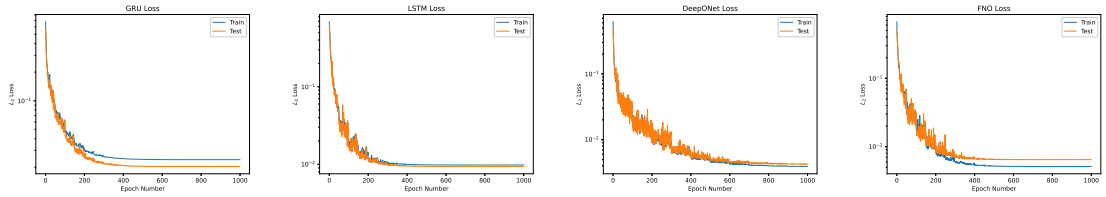

Figure 2: Loss functions for the training of each machine learning (ML) model architecture.

Additionally, we present 4 testing examples where we showcase each model's ability to mimic the original parameter estimator. These results are shown in 1.4.

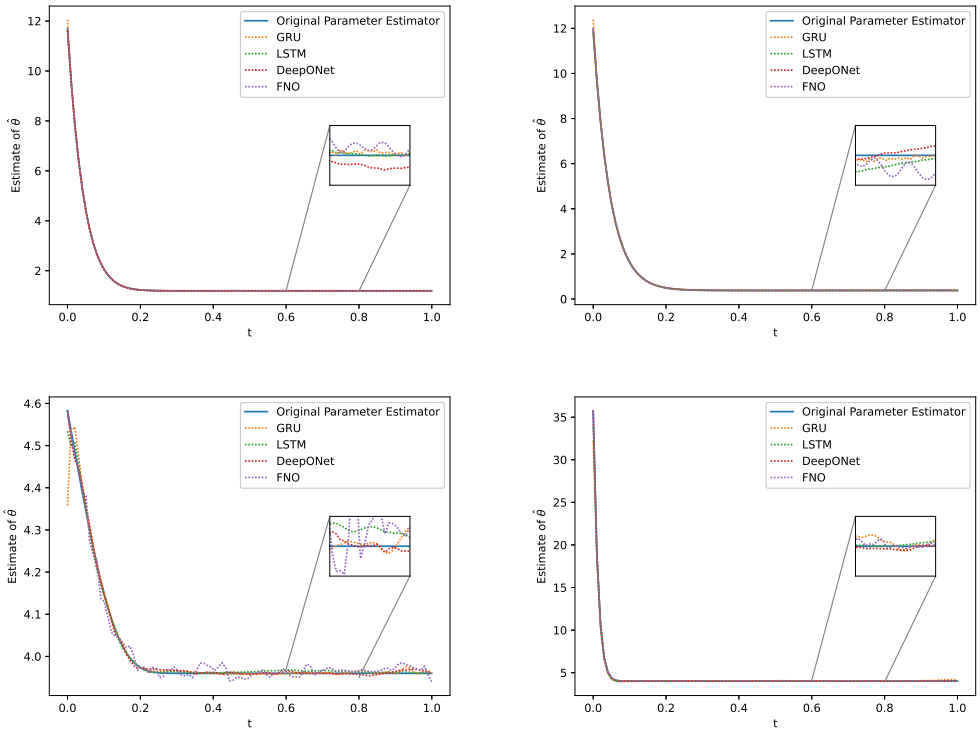

Figure 3: Example trajectories from the neural network models and the estimator for 4 different testing scenarios.

We can see that in all the cases, all 4 models approximate the original estimator exceptionally well except in the bottom left scenario where the FNO is much more jagged. However, we point out to the reader that the scale in the bottom left example is much smaller due to a good initial guess of α_0 for that scenario. The result of this smaller scale allows us to visualize that small errors encountered by each model and it can be shown that the other 3 scenarios have a similar behavior when zoomed. Lastly, we present a table with the overall training and testing results as well as the number of parameters and training time for each system in Table 2. Overall, all the methods perform quite well with the neural operators outperforming the RNN approaches slightly.

Model	**Parameter Size**	**Training Time (s/epoch)**	**Average Relative Error Training Data**	**Average Relative Error Testing Data**
GRU	485201	0.169	1.20E-3	1.01E-3
LSTM	646801	0.175	4.90E-4	4.68E-4
DeepONet	**98528**	**0.094**	**1.95E-4**	**2.12E-4**
FNO	107009	0.199	2.57E-4	3.24E-4

Table 2: Training and testing set results for each ML model. All experiments are run on an Nvidia RTX 3090.

2. Experimental System II: Aircraft System

2.1. Problem Statement

We consider a model for the wing-rock of an aircraft from Krstic et al. (1995).

$$\ddot{\phi} = \theta_1 + \theta_2\phi + \theta_3\dot{\phi} + \theta_4|\phi|\dot{\phi} + \theta_5|\dot{\phi}|\dot{\phi} \tag{28}$$

where ϕ is the angle, $\dot{\phi}$ is the angular velocity and θ_n is a constant parameter of the system. We can rewrite the above equations in a parametric form and introduce the control variable $u \in \mathbb{R}$.

$$\dot{\phi} = p \tag{29}$$

$$\dot{p} = \varphi^{\mathrm{T}}(\phi, p)\theta + u \tag{30}$$

with

$$\theta = [\theta_1,\ \theta_2,\ \theta_3,\ \theta_4,\ \theta_5]^{\mathrm{T}} \tag{31}$$

$$\varphi(\phi, p) = [1,\ \phi,\ p,\ |\phi|p,\ |p|p]^{\mathrm{T}} \tag{32}$$

After rewriting Eqn. 28 in a parametric form, we can then use the least-squares estimator from Krstic (2009) (See reference for derivation details).

$$\hat{\theta} = \alpha + \Gamma \int_0^p \varphi(\phi, \sigma)d\sigma \tag{33}$$

$$\dot{\Gamma} = -\Gamma\varphi\varphi^T\Gamma \tag{34}$$

$$\dot{\alpha} = -\Gamma\varphi\varphi^T\alpha - \Gamma p \int_0^p \frac{\partial\varphi(\phi, \sigma)}{\partial\phi}d\sigma - \Gamma\varphi u \tag{35}$$

with the control law of the form:

$$u = y_r^{(n)} + \left(K - \frac{\lambda}{2}e_2^T P\right)\tilde{x} - \varphi^T\hat{\theta} \tag{36}$$

where $x_r(t) = [y_r(t), \dot{y}_r(t)]^T$ is the vector of the reference trajectory function $y_r(t)$ and its derivative. $\tilde{x} = x - x_r(t)$, $\lambda > 0$, $e_2 = [0, 1]^T$, and $K = [k_1, k_2]$ such that the polynomial $x^2 + k_2x + k_1$ is Hurwitz. Additionally, let P be a positive definite matrix that satisfies the following:

$$P(A + e_2K) + (A + e_2K)^T P = -Q \tag{37}$$

where $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$. Additionally, we can compute the integrals in Eqns. 33, 34, 35 as the following:

$$\int_0^p \varphi(\phi, \sigma)d\sigma = p\left[1,\ \phi,\ \frac{p}{2},\ \frac{|\phi|p}{2},\ \frac{|p|p}{3}\right]^{\mathrm{T}} \tag{38}$$

$$\int_0^p \frac{\partial\varphi(\phi, \sigma)}{\partial\phi}d\sigma = \left[0,\ p,\ 0,\ \frac{p^2}{2}\mathrm{sgn}\{\phi\},\ 0\right]^{\mathrm{T}} \tag{39}$$

We provde an example solution of the system and the control law in Fig. 2.1

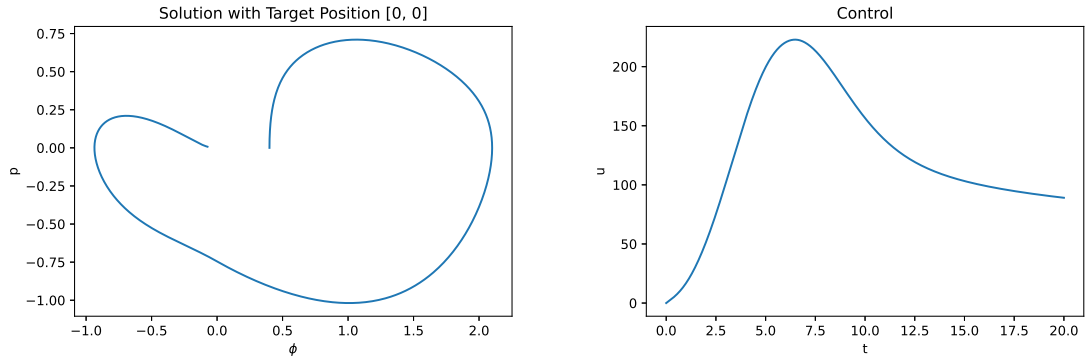

Figure 4: An example solution of the wing rock of an aircraft with the target function of $y_r(t) = 0$, $\theta = [0, -26.67, 0.76485, -2.9225, 0]$, $\alpha_0 = \theta * 1.25$, $\phi_{\text{init}} = [0.4, 0]$, $P = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$ and $\lambda = 1$.

2.2. Model Implementation and Hyperparameters

To build a dataset, we solve 1000 instances of the aircraft problem from $t \in [0, 20]$ and $\Delta t = 0.01$. We vary the following parameters again according to uniform distributions as $\theta = [\text{uniform}(0, 1), \text{uniform}(-24, -28), \text{uniform}(0.5, 1), \text{uniform}(-2, -4), \text{uniform}(-0.01, 0)]$, $x_{\text{init}} = [\text{uniform}(0, 1), 0]$, and $\alpha_0 = \theta * \text{uniform}(0, 2)$. Additionally, we note that under certain conditions, the wing rock model leads to numerical instability in the ordinary differential equation solver, but under the range listed above, the solution is always numerically computable at $\Delta t = 0.01$. We then aim to learn the mapping:

$$\begin{bmatrix} \phi(0) & \phi(\Delta t) & \dots & \phi(t=20) \\ p(0) & p(\Delta t) & \dots & p(t=20) \\ \alpha_0 & \alpha_0 & \dots & \alpha_0 \\ \alpha_1 & \alpha_1 & \dots & \alpha_1 \\ \alpha_2 & \alpha_2 & \dots & \alpha_2 \\ \alpha_3 & \alpha_3 & \dots & \alpha_3 \\ \alpha_4 & \alpha_4 & \dots & \alpha_4 \end{bmatrix} \implies \begin{bmatrix} \hat{\theta}(0) & \hat{\theta}(\Delta t) & \dots & \hat{\theta}(t=20) \end{bmatrix} \tag{40}$$

where α_n is the n^{th} element of the initial value of $\alpha = \alpha_{\text{init}}$ and stays constant throughout all time. Additionally, $\theta \in \mathbb{R}^5$ is the estimated paramters from Eqn. 33. We modify the architectures from the scalar plant slightly as the aircraft is certainly a harder problem to learn being multidimensional. We present the new architectures and hyper parameters in Table 3. Additionally, we utilize Gaussian Normalization for normalizing the data as it enhances the performance of all models.

Model	Network Architecture	Optimizer	Number of Parameters
Gated Recurrent Unit (GRU) Cho et al. (2014)	GRU(hiddenSize=850) $\mapsto$ Linear(width=1)	Adam(LR=0.01, $\beta = (0.9, 0.999)$, WeightDecay=0.0001) LRScheduler.stepLR(γ=0.5, stepSize=50)	2194705
Long Short-Term Memory (LSTM) Hochreiter and Schmidhuber (1997)	LSTM(hiddenSize=850) mapsto Linear(width=1)	Adam(LR=0.01, $\beta = (0.9, 0.999)$, WeightDecay=0.0001) LRScheduler.stepLR(γ=0.5, stepSize=50)	2924855
DeepONet Lu et al. (2021)	Branch: GRU(hiddenSize=200) $\mapsto$ Linear(width=500) Trunk: Linear(width=64) $\mapsto$ ReLU $\mapsto$ Linear(width=128) $\mapsto$ ReLU $\mapsto$ Linear(width=512) $\mapsto$ ReLU	Adam(LR=0.01, $\beta = (0.9, 0.999)$, WeightDecay=0.0001) LRScheduler.stepLR(γ=0.5, stepSize=50)	298848
Fourier Neural Operator (FNO) Li et al. (2020)	Linear (width=48) $\mapsto$ FFT (16 Modes) $\mapsto$ Linear (width=48) $\mapsto$ GeLU $\mapsto$ Linear (width=48) $\mapsto$ GeLU $\mapsto$ Linear (width=48) $\mapsto$ GeLU $\mapsto$ IFFT (16 Modes) $\mapsto$ Linear (width=1)	Adam(LR=0.01, $\beta = (0.9, 0.999)$, WeightDecay=0.0001) LRScheduler.stepLR(γ=0.5, stepSize=50)	311669

Table 3: Hyperparameters for various network architectures for the wing rock model of an aircraft. FFT and IFFT denote Fast Fourier Transform and Inverse Fast Fourier Transform respectively. LR denotes the Learning Rate.

2.3. Results

We again train each architecture on a 900/100 split for 100 epochs. We show the loss functions in Figure 2.3 and can see that the RNN based approaches converge much faster than those of the neural operator approaches, but also take much longer to train as shown in 4. Additionally, we show example solutions for estimating both $\hat{\theta}$ and using that estimator in the control scheme in Figures 2.3 and 2.3. It is important to note the scales on each figure as they are variable depending on the solution. This demonstrates our method is generalizable to different size problems and can accurately be applied in a control scheme. Overall summary statistics are presented in 3. We note that the speedup calculation does NOT involve the integration portion of the original parameter solver and therefore the speedup is most likely greater than the value presented here. We also note that the tensor operations are computed on a GPU while the ODE solver is on a CPU which is much slower. However, this is representative of common practice in control schemes and thus should be considered an additional justification for using a ML based model.

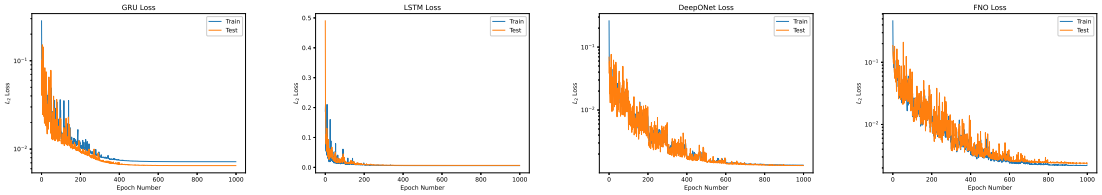

Figure 5: Loss functions for each learning model

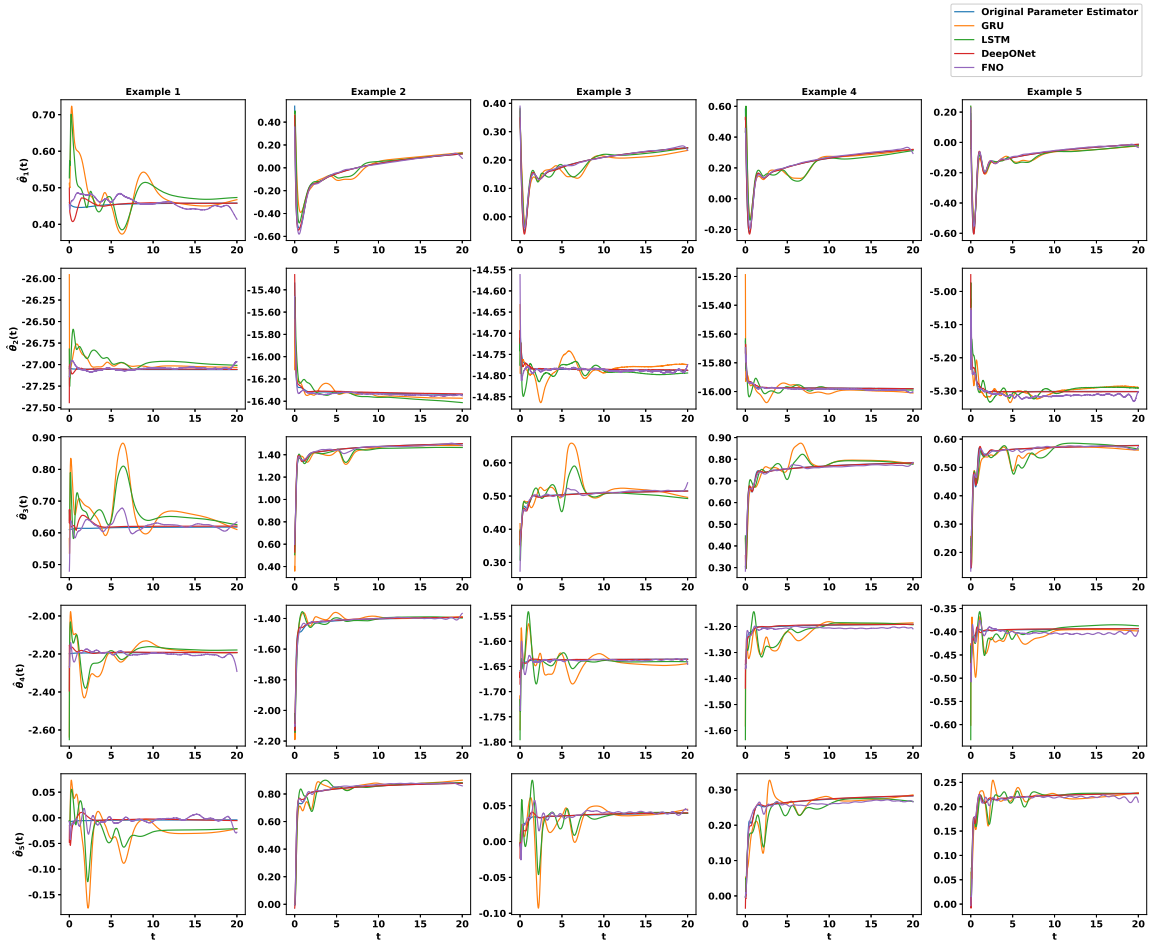

Figure 6: Parameter estimation estimates for 5 test instances. All examples correspond to the same instances as in Figure 2.3

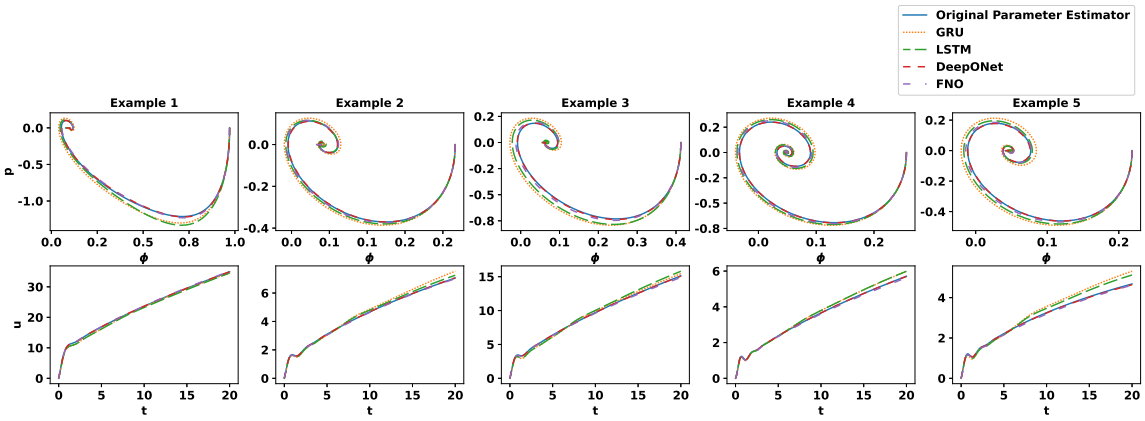

Figure 7: Replacement of the original estimator with each parameter estimator from Figure 2.3. Additionally, all the examples are the same instances as Figure 2.3.

Model	Parameter Size	Training Time (mins)	Average Relative Error Training Data	Average Relative Error Testing Data	Average Speed Up Over Original Parameter Estimator (%) CPU	Average Speed Up Over Original Parameter Estimator (%) GPU
GRU	2194705	87.33	7.20E-4	3.25E-4	-123	85.9
LSTM	2924855	98.49	6.58E-4	2.94E-4	-179	80.8
DeepONet	**298848**	36.78	**7.61E-5**	**5.98E-5**	29.4	92.7
FNO	311669	**9.41**	2.33E-4	1.06E-4	**94.6**	**94.9**

Table 4: Summary results for the wing-rock model. Green indicates the best value.

3. Experimental System III: Hyperbolic Partial Differential Equation (PDE) System

3.1. Problem Statement

We present an unstable hyperbolic PDE in the form:

$$u_t(x,t) = u_x(x,t) + \theta(x)u(0,t)\,, \tag{41a}$$

$$Y(t) = u(0,t)\,, \quad \text{(measured output)} \tag{41b}$$

$$u(1,t) = U(t)\,, \quad \text{(control input)} \tag{41c}$$

where $\theta(x) = 3\cos(\delta \arccos(x))$ is a Chebyshev polynomial of the form δ. We consider the following estimator from Bernard and Krstic (2014) that has been shown to stabilize the system such that $\lim_{t\to\infty} u(x,t) = 0$.

$$\hat{\theta}_t(x,t) = \gamma(x)\frac{Y(t-x)\left[Y(t) - U(t-1) - \int_{t-1}^{t}\hat{\theta}(t-\tau,t)Y(\tau)d\tau\right]}{1+\int_{t-1}^{t}Y^2(\tau)d\tau} \tag{42}$$

where U(t) is the control given by the following:

$$U(t) = \int_{t-1}^{t}\hat{\kappa}(t-\tau,t)U(\tau)d\tau + \int_{t-1}^{t}\left[\int_{t-\tau}^{1}\hat{\kappa}(\mu,t)\hat{\theta}(1-\mu+t-\tau,t)d\mu\right]Y(\tau)d\tau \tag{43}$$

and $\hat{\kappa}(x,t)$ is known as the gain function defined as:

$$\hat{\kappa}(x,t) = -\hat{\theta}(x,t) + \int_0^x \hat{\kappa}(y,t)\hat{\theta}(x-y,t)dy \tag{44}$$

We showcase examples of the solutions in Figure 3.1. From the examples, we can see that this is a very complex dataset with multiple functional forms to be learned.

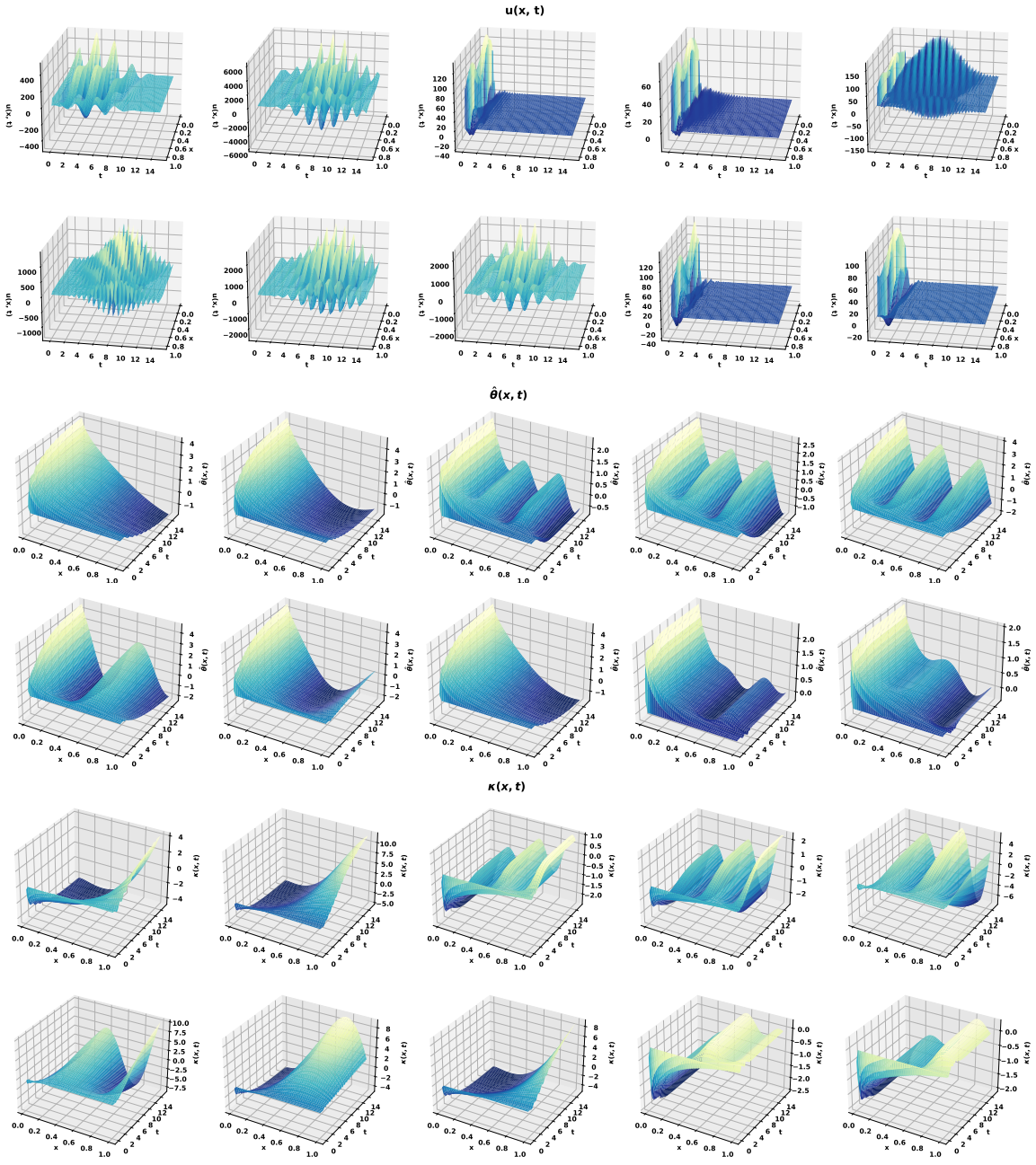

Figure 8: Example solutions of u, $\hat{\theta}$ and κ with various Chebyshev inputs.

3.2. Parameter Estimator

3.2.1. DATASET GENERATION AND HYPERPARAMETERS

To build a dataset, we solve 1000 instances of the PDE from $t \in [0, 15]$, $\Delta t = 0.01$ and $x \in [0, 1]$, $\Delta x = 0.01$. We vary the following parameters according to uniform distributions $\delta = \text{uniform}(1, 12)$, $u(0, x) = \text{uniform}(50, 100)$, and
$\alpha(0, x) = 0$. δ controls the type of the Chebyshev function and creates vastly different functions as shown in Figure 3.1. This choice of $\theta(x)$ is very important for a rich and interesting dataset as many choices for $\theta(x)$ such as $\sin(x)$ are very very easy to learn since they do not create a wide array of parameter estimation and gain functions.

$$\begin{bmatrix} u(0,0) & u(1,0) \\ u(0,\Delta t) & u(x=1,\Delta t) \\ \vdots & \vdots \\ u(0,t=20) & u(x=1,t=20) \end{bmatrix} \mapsto \begin{bmatrix} \hat{\theta}(0,0) & \hat{\theta}(\Delta x,0) & ... & \hat{\theta}(x=1,0) \\ \hat{\theta}(0,\Delta t) & \hat{\theta}(\Delta x,\Delta t) & ... & \hat{\theta}(x=1,\Delta t) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\theta}(0,t=20) & \hat{\theta}(\Delta x,t=20) & ... & \hat{\theta}(x=1,t=20) \end{bmatrix} \tag{45}$$

where α is constant at $t = 0$. Considering the large size of the PDE, we down sample to $\Delta x = 0.02$ and $\Delta t = 0.05$ when feeding the dataset to the neural network models. We emphasize the PDE is still solved at the smaller time and spatial steps. It is also worth noting it takes around 2 hours and 30 minutes to generate the dataset. To train the models, we follow a similar approach as the aircraft model; however, since we have a 2D problem, we change the projection layers of the models slightly as available in Table 5.

Model	Network Architecture	Optimizer	Number of Parameters
Gated Recurrent Unit (GRU) Cho et al. (2014)	GRU(hiddenSize=500) $\mapsto$ Linear(width=50)	Adam(LR=0.01, $\beta = (0.9, 0.999)$, WeightDecay=0.0001) LRScheduler.stepLR(γ=0.5, stepSize=50)	853050
Long Short-Term Memory (LSTM) Hochreiter and Schmidhuber (1997)	LSTM(hiddenSize=500) mapsto Linear(width=50)	Adam(LR=0.01, $\beta = (0.9, 0.999)$, WeightDecay=0.0001) LRScheduler.stepLR(γ=0.5, stepSize=50)	1129050
DeepONet Lu et al. (2021)	Branch: GRU(hiddenSize=200) $\mapsto$ Linear(width=500) Trunk: Linear(width=512) $\mapsto$ ReLU $\mapsto$ Linear(width=500) $\mapsto$ ReLU $	Adam(LR=0.01, $\beta = (0.9, 0.999)$, WeightDecay=0.0001) LRScheduler.stepLR(γ=0.5, stepSize=50)	534312
Fourier Neural Operator (FNO) Li et al. (2020)	Linear (width=50) $\mapsto$ FFT (16 Modes) $\mapsto$ Linear (width=32) $\mapsto$ GeLU $\mapsto$ Linear (width=32) $\mapsto$ GeLU $\mapsto$ Linear (width=32) $\mapsto$ GeLU $\mapsto$ IFFT (16 Modes) $\mapsto$ Linear (width=50)	Adam(LR=0.01, $\beta = (0.9, 0.999)$, WeightDecay=0.0001) LRScheduler.stepLR(γ=0.5, stepSize=50)	427954

Table 5: Hyperparameters for various network architectures for the unstable hyperbolic PDE. FFT and IFFT denote Fast Fourier Transform and Inverse Fast Fourier Transform respectively. LR denotes the Learning Rate.

3.2.2. RESULTS

We show results for only the parameter estimation here. We can see that all the neural network approaches converge in Fig 3.2.2, but the FNO architecture performs much better than the other architectures in terms of error in Table 6. Additionally, we present an example of the estimator and

various errors at the boundary in Figure 3.2.2 and the control employing each estimator in Figure 3.2.2. We can see that small perturbations in the estimation can translate to much larger errors in the control, but we believe this is due to a phase shift of the controllers. Additionally, we can see that in all neural network estimation emulators, the system still stabilizes without issues. Furthermore, we can see that all approaches get very similar performance speedups on the GPU, but the FNO performs best on the CPU due to the fact there is no internal state. (The speedups are calculated by taking the average of the times for 5 $\hat{\theta}$ samples in Matlab and then comparing that to the time for the ML to perform a calculation from the system measurements to $\hat{\theta}$). Lastly, we can see the ML approaches all require many less parameters in this approach due to the sub-sampling when compared to the aircraft system. In practice, this approach of sampling is more reasonable then the 100 Hertz sampling in the aircraft system especially for multi-dimensional problems.

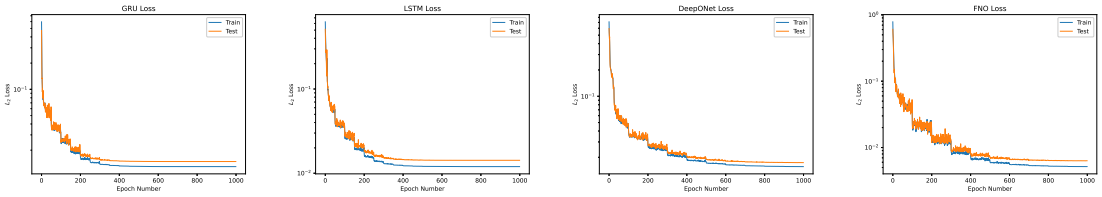

Figure 9: Loss functions for each neural network model

Model	Parameter Size	Training Time (mins)	Average Relative Error Training Data	Average Relative Error Testing Data	Average Speed Up Over Original Parameter Estimator (%) CPU	Average Speed Up Over Original Parameter Estimator (%) GPU
GRU	853050	8.24	6.4E-4	7.41E-4	-8.99	**96.6**
LSTM	1129050	8.23	5.9E-4	7.12E-4	-48.9	**96.6**
DeepONet	534312	8.36	7.8E-4	8.70E-4	-16.8	96.3
FNO	**427954**	**4.42**	**2.5E-4**	**3.16E-4**	**90.05**	91.3

Table 6: Summary Results for the PDE Estimator. Green denotes the best value.

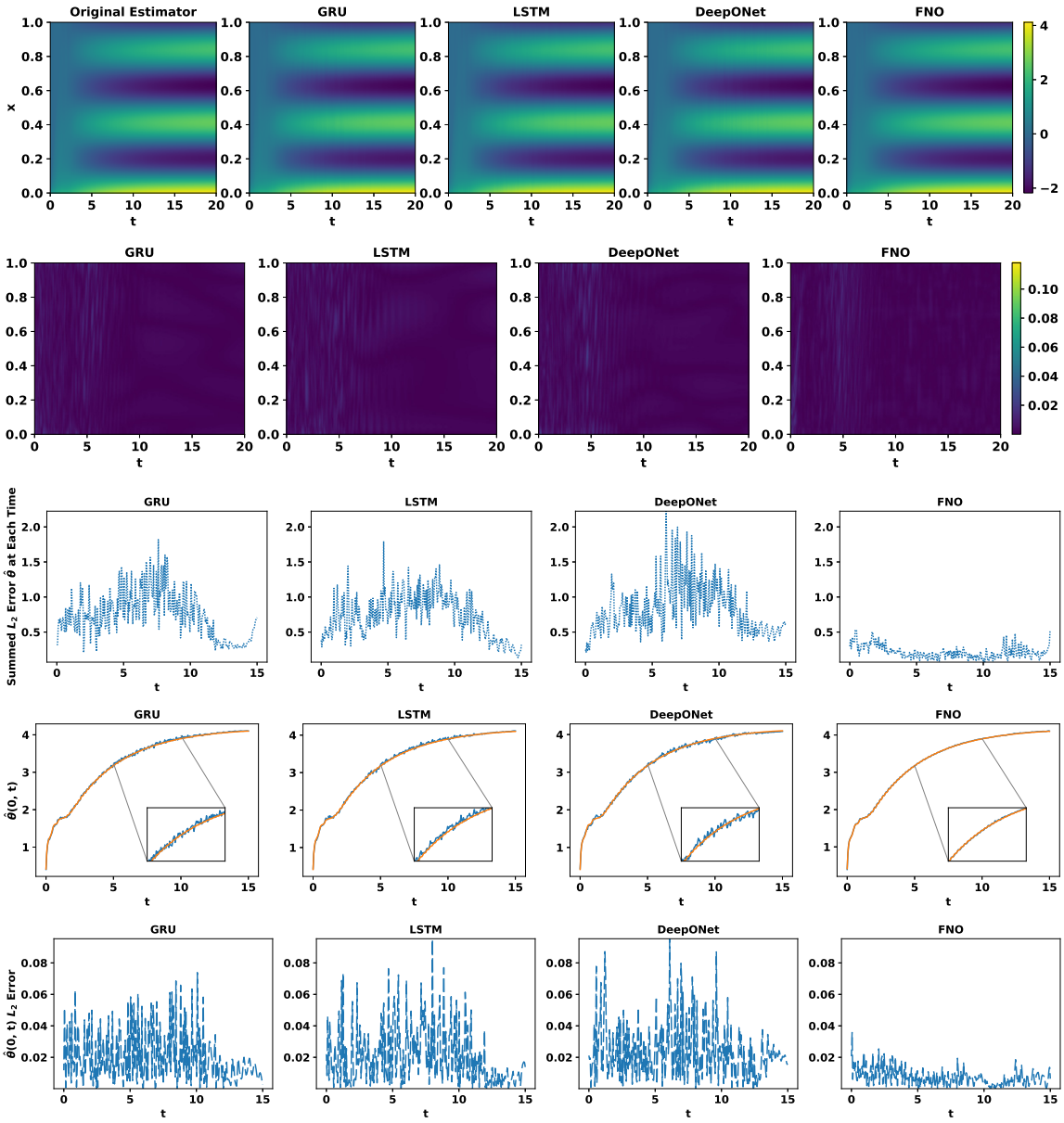

Figure 10: Example of PDE Estimator $\hat{\theta}$ with each learning approach. Let $\tilde{\theta}$ be the estimation from the learning based approaches and $\hat{\theta}$ be the estimates from the original estimator. Displayed from to to bottom are the estimator $(\hat{\theta}(x,t), \tilde{\theta}(x,t))$, the estimator's absolute error $(|\hat{\theta}(x,t) - \tilde{\theta}(x,t)|)$, the summed absolute estimation error at each timestep $(\int_0^1 |\hat{\theta}(\sigma,t) - \tilde{\theta}(\sigma,t)|d\sigma)$, the ML estimator in blue overlayed with the original estimator in orange at the boundry$(\tilde{\theta}(0,t)$, $\hat{\theta}(0,t))$, and the estimator boundry absolute error $(|\hat{\theta}(0,t) - \tilde{\theta}(0,t))$.

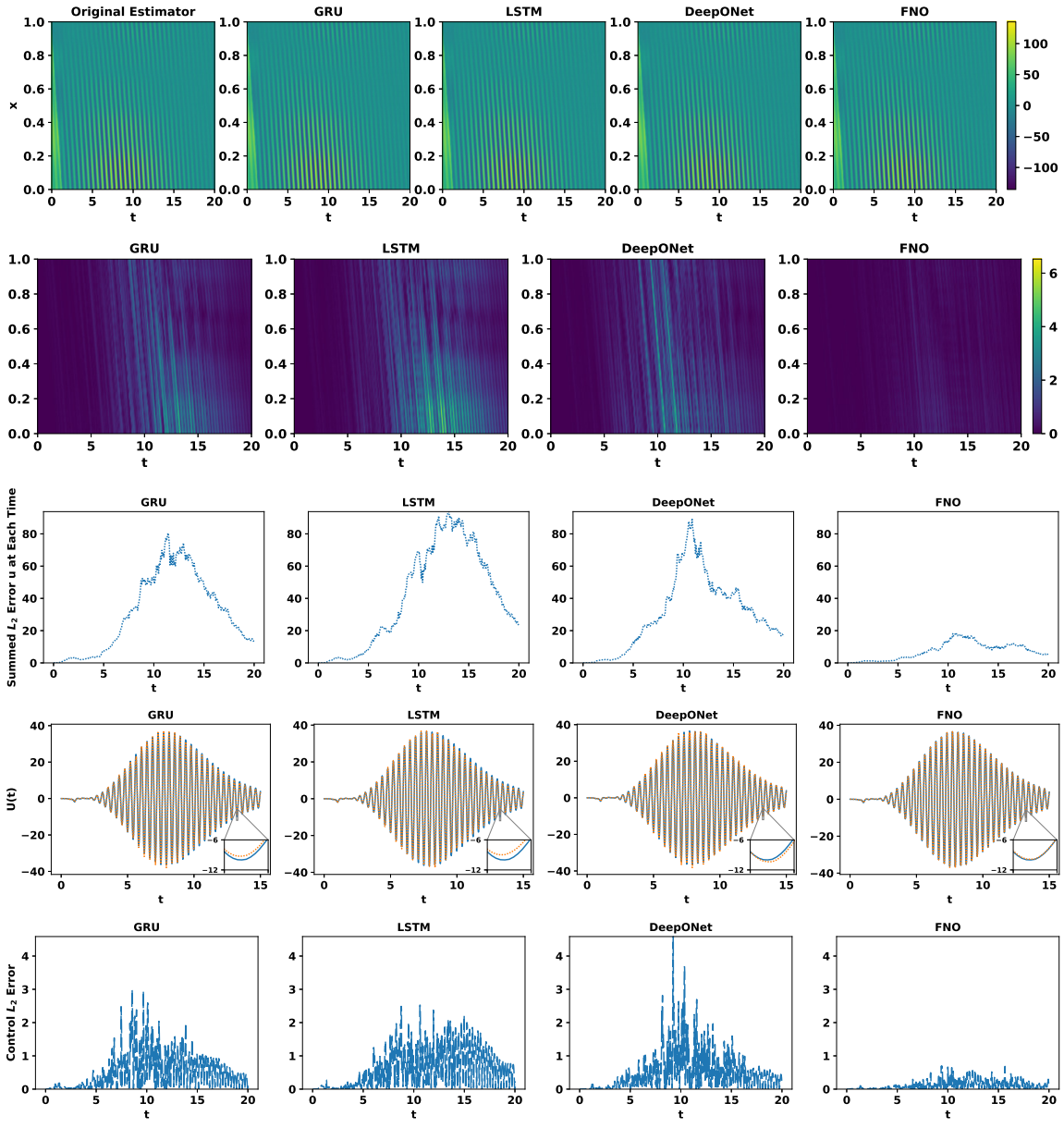

Figure 11: Example of PDE control with the replaced parameter estimator. Displayed from top to bottom are the solution ($\hat{u}(x,t)$), the solution's absolute error ($|\hat{u}(x,t) - u(x,t)|$), the summed solution error at each timestep ($\int_0^1 \hat{u}(\sigma,t) - u(\sigma,t)d\sigma$), the ML estimator based control in orange overlayed with the original estimator in , and the control error ($\hat{U}(t) - U(t)$).

3.3. Gain Function

3.3.1. DATA GENERATION AND HYPERPARAMETERS

Again using the dataset mentioned in Section 3.2.1, we attempt to learn the gain function. It is worth noting that the gain function is not time dependent and so we sample 5 timesteps for each of the 1000 instances to create a dataset of 5000 instances. We then learn the mapping

$$\begin{bmatrix}\hat{\theta}(0) & \hat{\theta}(\Delta x) & \hat{\theta}(2\Delta x) & \dots & \hat{\theta}(x=1)\end{bmatrix} \mapsto \begin{bmatrix}\kappa(0) & \kappa(\Delta x) & \kappa(2\Delta x) & \dots & \kappa(x=1)\end{bmatrix} \quad (46)$$

To train the models in only 1D, we can use a much simpler set of models and obtain the following hyperparameters shown in Table 7.

Model	Network Architecture	Optimizer	Number of Parameters
Gated Recurrent Unit (GRU) Cho et al. (2014)	GRU(hiddenSize=500)	Adam(LR=0.01, $\beta = (0.9, 0.999)$, WeightDecay=0.0001) LRScheduler.stepLR(γ=0.5, stepSize=50)	853050
Long Short-Term Memory (LSTM) Hochreiter and Schmidhuber (1997)	LSTM(hiddenSize=500)	Adam(LR=0.01, $\beta = (0.9, 0.999)$, WeightDecay=0.0001) LRScheduler.stepLR(γ=0.5, stepSize=50)	1129050
DeepONet Lu et al. (2021)	Branch: GRU(hiddenSize=200) $\mapsto$ Linear(width=64) Trunk: Linear(width=32) $\mapsto$ ReLU $\mapsto$ Linear(width=64) $\mapsto$ ReLU	Adam(LR=0.01, $\beta = (0.9, 0.999)$, WeightDecay=0.0001) LRScheduler.stepLR(γ=0.5, stepSize=50)	534312
Fourier Neural Operator (FNO) Li et al. (2020)	Linear (width=100) $\mapsto$ FFT (16 Modes) $\mapsto$ Linear (width=32) $\mapsto$ GeLU $\mapsto$ Linear (width=32) $\mapsto$ GeLU $\mapsto$ Linear (width=32) $\mapsto$ GeLU $\mapsto$ IFFT (16 Modes) $\mapsto$ Linear (width=50)	Adam(LR=0.01, $\beta = (0.9, 0.999)$, WeightDecay=0.0001) LRScheduler.stepLR(γ=0.5, stepSize=50)	427954

Table 7: Hyperparameters for various network architectures for the gain function of the PDE. FFT and IFFT denote Fast Fourier Transform and Inverse Fast Fourier Transform respectively. LR denotes the Learning Rate.

3.3.2. RESULTS

We showcase converged loss functions in Fig 3.3.2. Additionally, we consider the gain function for entire PDE solutions across time. Thus, in Figures 3.3.2 and 3.3.2 the learned gain function is applied at each timestep and the results are shown over the entire solution. Furthermore, we can see the results for the gain function's entire dataset in Table 8.

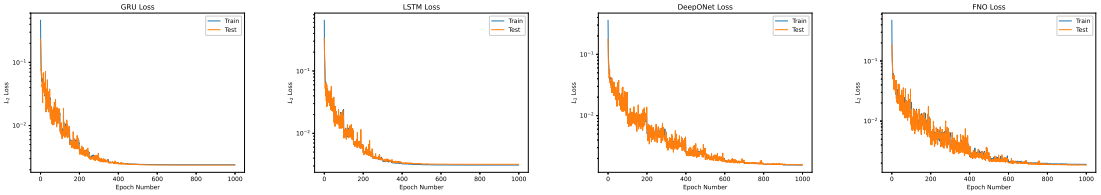

Figure 12: Loss functions for learning the gain κ

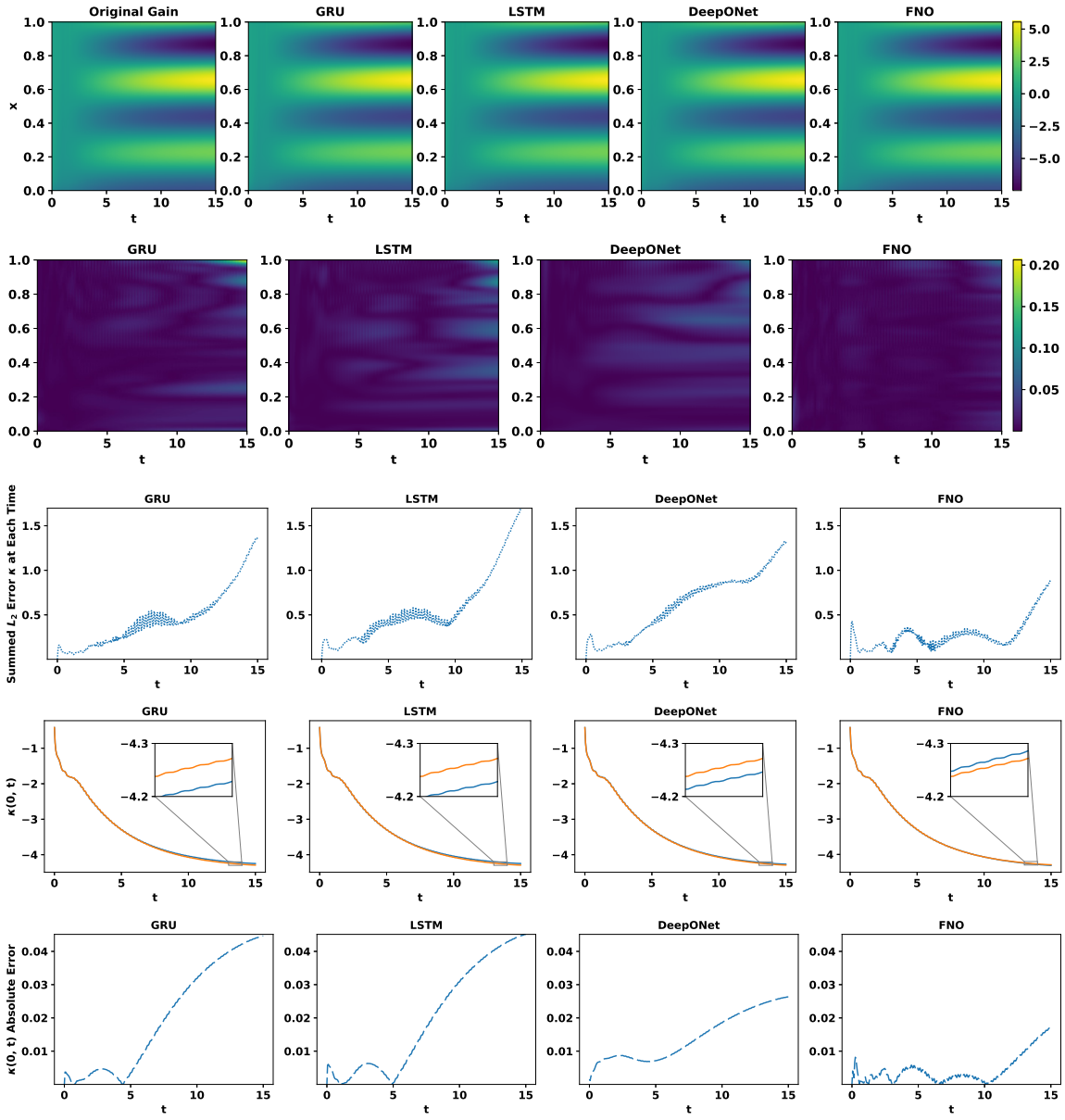

Figure 13: Example of gain function $\kappa(\hat{\theta}, x)$ with each learning approach. Let $\tilde{\kappa}(\hat{\theta}, x)$ be the gain function from the learning based approaches and κ be the original gain function. Displayed from to to bottom are the gain function $(\kappa(\hat{\theta}, x), \tilde{\kappa}(\hat{\theta}, x))$, the gain's absolute error $(|\kappa(\hat{\theta}, x) - \tilde{\kappa}(\hat{\theta}, x)|)$, the summed absolute gain function error at each timestep $(\int_0^1 |\kappa(\hat{\theta}, \sigma) - \tilde{\kappa}(\hat{\theta}, \sigma)| d\sigma)$, the ML gain function in blue overlayed with the original gain function in orange at the boundry$(\tilde{\kappa}(\hat{\theta}, 0)$, $\kappa(\hat{\theta}, 0))$, and the estimator boundry absolute error $(|\kappa(\hat{\theta}, 0) - \tilde{\kappa}(\hat{\theta}, 0))$.

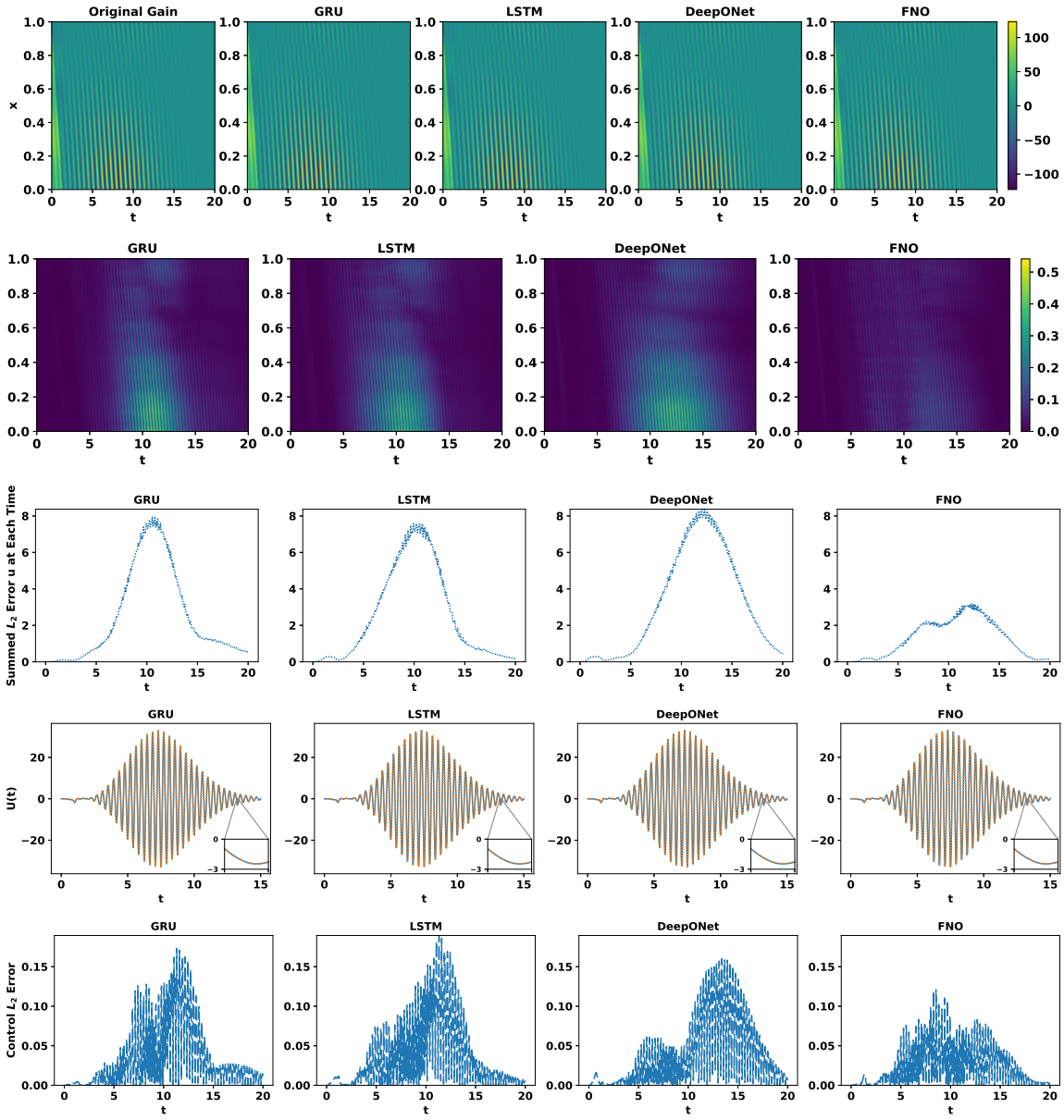

Figure 14: Example of PDE control with the replaced gain function. Displayed from top to bottom are the solution ($\hat{u}(x,t)$), the solution's absolute error ($|\hat{u}(x,t) - u(x,t)|$), the summed solution error at each timestep ($\int_0^1 \hat{u}(\sigma,t) - u(\sigma,t)d\sigma$), the ML replaced gain function based control in orange overlayed with the original gain function's control in , and the control error ($\hat{U}(t) - U(t)$).

Model	Parameter Size	Training Time (mins)	Average Relative Error Training Data	Average Relative Error Testing Data	Average Speed Up Over Original Parameter Estimator CPU (%)	Average Speed Up Over Original Parameter Estimator GPU (%)	Average Calculation Time CPU (sec)	Average Calculation Time GPU(sec)
GRU	755001	3.99	1.99E-4	2.00E-4	73.3	**98.9**	0.863	0.0344
LSTM	1006501	3.96	2.76E-4	2.76E-4	63.6	**98.9**	1.18	0.0344
DeepONet	136840	10.47	2.7E-4	2.69E-4	72.3	98.6	0.894	0.0425
FNO	**106977**	**7.13**	**1.83E-4**	**1.80E-4**	**94.7**	96.8	0.170	0.104

Table 8: Summary of PDE gain function results.

3.4. Combining Both Learned Parameter Estimator and Gain Function with Controller

Figure 3.4 demonstrates the control when both the estimator and the gain function are replaced by learning-based approaches. The parameter estimation outputs are fed into the gain function in this scenario, and therefore the entire portion of estimation and gain function is completely machine learning based. We can see that there is very little error increase over replacing the function with just the estimator which is excellent since the gain function gives a very large speedup. Furthermore, we can see that all neural network approaches provide speedup, and depending on the system architecture, some approaches provide better speed-up over other approaches.

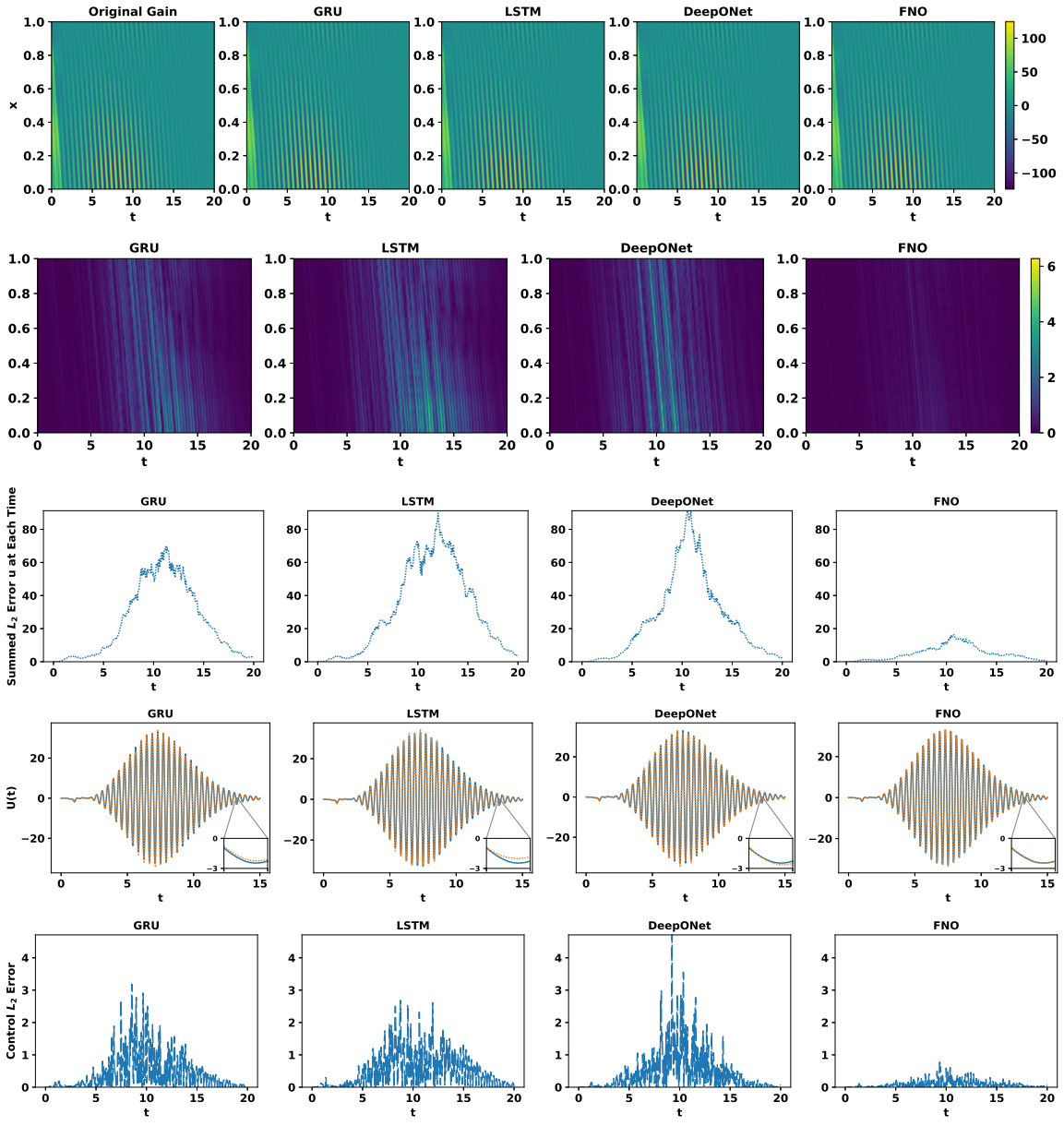

Figure 15: Example of PDE control with the replaced gain function and parameter estimator. Displayed from top to bottom are the solution ($\hat{u}(x,t)$), the solution's absolute error ($|\hat{u}(x,t)-u(x,t)|$), the summed solution error at each timestep ($\int_0^1 \hat{u}(\sigma,t)-u(\sigma,t)d\sigma$), the ML replaced gain function based control in orange overlayed with the original gain function's control in , and the control error ($\hat{U}(t)-U(t)$).

Model	Combined Parameter Size	Combined Training Time (mins)	Average Relative Error Testing Data For Solution u	Average Relative Error Testing Data For Control U	Average Speed Up Over Original Parameter Estimator CPU (%)	Average Speed Up Over Original Parameter Estimator GPU (%)	Average Calculation Time CPU (sec)	Average Calculation Time GPU(sec)
GRU	1608051	12.23	8.96E-4	3.20E-3	72.5	**98.2**	0.89	**0.057**
LSTM	2135551	12.19	8.12E-4	3.19E-3	62.1	**98.2**	1.23	0.059
DeepONet	671152	18.83	9.9E-4	3.79E-3	72.9	98.0	0.875	0.065
FNO	**534931**	**11.55**	**4.46E-4**	**1.52E-3**	**92.5**	92.7	**0.242**	0.237

Table 9: Summary of PDE with both gain function and estimator replaced.

Acknowledgments

We thank Professor Paulina Bernard for her code on the hyperbolic PDE system, and Jie Feng, Zongyi Li for interesting conversations surrounding hyperparameters and learning techniques.

References

Pauline Bernard and Miroslav Krstic. Adaptive output-feedback stabilization of non-local hyperbolic pdes. *IFAC Proceedings Volumes*, 47(3):7755–7760, 2014. ISSN 1474-6670. doi: https://doi.org/10.3182/20140824-6-ZA-1003.00108. URL https://www.sciencedirect.com/science/article/pii/S147466701642834X. 19th IFAC World Congress.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches, 2014. URL https://arxiv.org/abs/1409.1259.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Petros A. Ioannou and Jing Sun. *Robust Adaptive Control*. Prentice-Hall, Inc., USA, 1995. ISBN 0134391004.

Miroslav Krstic. On using least-squares updates without regressor filtering in identification and adaptive control of nonlinear systems. *Autom.*, 45(3):731–735, 2009. doi: 10.1016/j.automatica.2008.09.024. URL https://doi.org/10.1016/j.automatica.2008.09.024.

Miroslav Krstic, Petar V Kokotovic, and Ioannis Kanellakopoulos. *Nonlinear and adaptive control design*. John Wiley & Sons, Inc., 1995.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2020. URL https://arxiv.org/abs/2010.08895.

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. doi: 10.1038/s42256-021-00302-5. URL https://doi.org/10.1038%2Fs42256-021-00302-5.