TOWARDS AN OPERATIONALLY MEANINGFUL, EXPLAINABLE EMULATOR FOR THE BOUSSINESQ EQUATION

Anonymous authors
Paper under double-blind review

ABSTRACT

The demand for emulators of expensive computational models is rapidly surging, as search, optimization, and uncertainty quantification tasks become increasingly more relevant than the computation of single simulation configurations. Along with traditional methods that explore a model's configuration space efficiently and accurately, relatively more recent AI algorithms have started to play a role in the construction of simulation emulators. In particular, Auto-Encoder networks have been deployed in a variety of domains, as they not only can ingest simulation data and enable the fast evaluation of specific observables, but also provide a degree of transparency into the evaluation process and relate the predicted values to the model's configuration variables through simple, low-dimensional representations. We apply a recent proposal, based on a modified Variational Auto-Encoder (VAE) architecture, to a 2D coastal inundation model based on the Boussinesq equation, and evaluate its ability to learn the relationship between the model controls and the observed water levels at a predefined location. We also propose two routes to use this representation as the central component of an emulator, and assess their respective viability.

1 INTRODUCTION

A computational model is effectively a map between a configuration space, spanned by parameters such as the model's initial conditions, domain geometry, or physical parameters, and an observable space, consisting of the variables of interest which the model aims to compute. The map can be complex and computationally expensive, so that in practice only a limited portion of the configuration space can be probed, and inverting the map is rarely tractable in practice.

There is, however, hope for improvement: whilst the dimensionality and coordinates of the configuration space are usually motivated by physics and respond to the need to model specific properties of the simulation system, the representation of this space thus obtained is by no means guaranteed to be the most compressed, or decoupled. Symmetries and conservation laws, as well as couplings between degrees of freedom and cancellations in the governing equations, may reduce the effective dimensionality of the configuration space for the purpose of the computation of specific observables, and lead to simpler representations thereof. This fact is of particular interest when the model is only used to compute specific observable values, as an overtly complicated, redundant representation can compromise and mislead the search task.

Several past efforts have illustrated strategies to extract more concise physics descriptions from datasets (not necessarily originating from computational modelling). These efforts adopt either a sparse-regression approach to reconstruct the governing equations for a given dataset (Schmidt & Lipson, 2009; Brunton et al., 2016; Rudy et al., 2017), or attempt to use standard ML techniques for data compression (such as VAE networks) to discover the basic parameters that control a dataset (Iten et al., 2020; Nautrup et al., 2020). For datasets produced by simulation models, where the equations governing the evolution of the data are already known, the latter approach seems much more promising, and is the one we will adopt in this paper.

In the following, we will show how a network of the type proposed in Iten et al. (2020); Nautrup et al. (2020) can be trained on a dataset consisting of several thousands of coastal-dynamics simulations.

We will then assess the network's ability to predict new dynamics, and examine the resulting latent representation. As we tackle both of these tasks, we will attempt to answer the key question of how an emulator might be built at the end of this procedure.

2 SIMULATION MODEL

The model used for all simulations performs the evolution of water-height data over a 2D domain, according to the Boussinesq equation. The model is implemented in the Cactus software framework (Loffler et al., 2012), as described in Oler et al. (2016); Chakrabarti et al. (2017). The basic setup involves a two-level coastal profile, representing land and sea on either side of a straight stretch of coast, shielded in part by a straight, infinite-height levee parallel to the coast. The water height's initial configuration is that of a plane wave, also parallel to the coast. The simulation computes the evolution of the water level everywhere on the domain, as the initial profile marches inwards towards the coast and waves are scattered around the domain. The model specification is summarised in Table 2 within Section 2, along with sample plots of the wave configurations and evolution spanned by our dataset.

We construct a dataset of 12,006 simulations by perturbing this basic setup. The variation consists in a parametrized deformation of the levee, modelled by a sinusoidal displacement in the x direction with amplitude a and wave number n:

$$x_L = \bar{x}_L + a \sin(\pi n y_L) \tag{1}$$

The amplitude a is sampled between -10 and 10, with a stride of 0.01, while the wave number n is in $\{1, 6\}$. The key observable computed during the simulation is the maximum water level $d_\star$ reached by the water at a point immediately behind the levee, $x_\star = 40, y_\star = 25$. It is clear, from the plots in Section 2 as well as from basic fluid-dynamics arguments, that the relationship between the levee morphology and the maximum water level behind the levee is inevitably non-linear.

3 RESULTS

The main question we aim to explore in this paper is whether it is feasible to learn the map $f : (a, n) \to d_\star$, so as not only to reconstruct f for any pair (a, n) (within a certain domain and up to a certain accuracy), but also to capture and make physical sense of the key features of f.

An interesting approach to this end, described in Iten et al. (2020); Nautrup et al. (2020), involves the use of a modified β Variational Auto-Encoder (β-VAE) architecture (called *SciNet*) to discover the minimum-dimensional representation of the parameter space spanned by a dataset, given a specific observable of interest. In other words, the strategy aims to discover not the most generic and minimal representation for the full phenomenology described by a dataset, but only that subset of this representation which is useful to predict specific observable properties. Such representations are dubbed *operationally meaningful*. Statistical independence of the individual components of this representation is encouraged by the variational approach; the relative importance of reconstruction accuracy and statistical correlation can be tuned through the parameter β, which weighs the contribution of the latter term in the β-VAE's loss function.

As our problem closely mirrors this structure, in that it is not the general coastal wave dynamics we are interested in, but merely the resulting water level in a specific point of the domain, the particular scheme proposed in Iten et al. (2020); Nautrup et al. (2020) appears promising.

We are immediately faced with a problem. *SciNet*'s method is designed to learn representations in generic data-driven scenarios. Once a network is trained on a set of observations, a new prediction can be obtained by feeding the β-VAE a new observation and letting it infer the target quantity. In our case, however, the direct production of a new observation is precisely the expensive step we are attempting to circumvent. A trained emulator would ideally take the values of the model's input parameters (not the model's observed properties, which would require a costly simulation), and, based on those alone, predict the target observable.

We envisage two possible modifications to *SciNet*'s approach that would address this type of application:

Table 1: Training results for various values of the network parameters. All networks were trained for 50 epochs. The last three columns represent the mean result over ten random train/test partitions. The digits in parentheses indicate the order of magnitude of the standard deviation.

Configuration	N_{train}	n_{latent}	n_{layer}	β	Loss	Reconstruction RMSE	Prediction RMSE
a	10006	2	64	0	$1.(8) \cdot 10^{-4}$	$(6) \cdot 10^{-4}$	$(6) \cdot 10^{-5}$
b	10006	2	256	0	$(2) \cdot 10^{-4}$	$(6) \cdot 10^{-4}$	$2.(1) \cdot 10^{-5}$
c	10006	2	1024	0	$(3) \cdot 10^{-3}$	$(2) \cdot 10^{-3}$	$(5) \cdot 10^{-5}$
d	10006	1	64	0	$(5) \cdot 10^{-4}$	$(8) \cdot 10^{-4}$	$3.(0) \cdot 10^{-5}$
e	10006	5	64	0	$(6) \cdot 10^{-3}$	$(2) \cdot 10^{-3}$	$(7) \cdot 10^{-5}$
f	10006	10	64	0	$(2) \cdot 10^{-3}$	$1.(9) \cdot 10^{-3}$	$(6) \cdot 10^{-5}$
g	10006	2	64	0.2	$-(8) \cdot 10^{-2}$	$(2) \cdot 10^{-3}$	$(7) \cdot 10^{-5}$
h	10006	2	64	0.8	$-3.(9) \cdot 10^{-1}$	$1.(2) \cdot 10^{-3}$	$4.(1) \cdot 10^{-5}$
i	11886	2	64	0	$(3) \cdot 10^{-4}$	$(7) \cdot 10^{-4}$	$(8) \cdot 10^{-5}$

- **Approach I:** only the decoder part of the trained network is used, so that one need only specify values for the latent neurons to obtain a prediction. This approach, however, is predicated on an important assumption: that the latent representation has a clear, explainable relationship with the original parameters (in our case, a and n), so that given values of a and n, it is straightforward to compute the corresponding values for the latent neurons. We will test this assumption in this paper, but will reserve for a future study the formulation of a method that guarantees that this assumption is indeed satisfied.
- **Approach II:** the second solution is to attempt training the network with observations of the system's initial configuration, before any expensive evolution has to happen. This is the approach we will use in this paper.

We construct a dataset with 12,006 entries representing the levee shape (essentially, equation (1), for $(a, n) \in [-10, 10] \times \{1, 6\}$, as described in section 2. The shape is discretized as a vector of size 60, and each entry is labelled by the maximum water level $d_\star$ at $x_\star = 40m, y_\star = 25m$. We train a β-VAE with a 60-neuron input layer, a 1-neuron output layer, a two-level encoder and decoder of size n_{layer}, and a latent representation of size n_{latent} (the architecture corresponds to Figure 1(b) in Iten et al. (2020), except for the question neuron, which we omit). We illustrate the result of this procedure, for various values of n_{latent}, n_{layer}, and β, in Table 1. For each configuration described in the table, we perform the training 10 times: each time, we partition the dataset by randomly extracting N_{train} simulations to use for the training, and using the remaining 12,006$-N_{\text{train}}$ entries for testing. The results in the last three columns represent the mean of each set of ten experiments (the digit in parentheses indicating the order of magnitude of the standard deviation). Plots of the different datasets and the corresponding network predictions are shown in Section B.

The method performs reasonably well on small deformation amplitudes, yielding mean prediction RMSEs of around 0.01%, whilst the dataset variance is around two orders of magnitude higher. Some basic properties that control the variation of $d_\star$ are also captured correctly, such as the fact that $d_\star$ generally increases with $|a|$, and that odd values of n lead to larger $d_\star$. The network's performance on a sample dataset partition is shown in Figure 1, obtained for the first configuration shown in the table. The complete results of this procedure for all datasets are shown in section B.

The results are very promising: we can presume that the dependence of $d_\star$ on a and n is highly non-trivial, corresponding to non-linear fluid-dynamic effects which couple wave evolution and domain geometry in a way that is difficult to forecast in a simple, generalizable way. Yet, our approach can both capture this dependence accurately, and provide a strategy that extends trivially to other geometries, initial wave profiles, and even physical parameters and equations. We do need to note, however, that the quality of both reconstruction and prediction appears to degrade for the simulations where the water levels are particularly high (those corresponding to high $|a|$); an interesting follow-up question is how well this method would apply to scenarios with extreme outliers, as we could perhaps expect from models with more complex geometries or higher-dimensional configuration spaces.

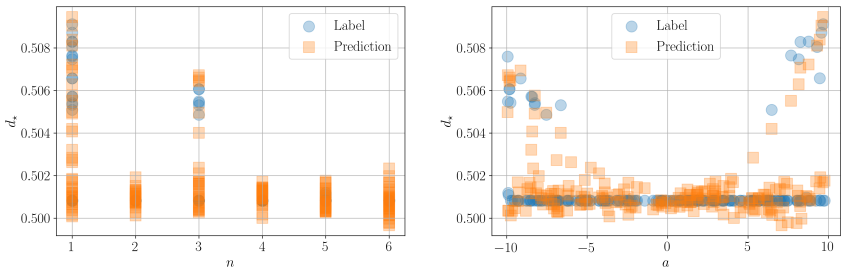

Figure 1: Inference results for one of the dataset partitions with $N_{\text{train}} = 10006$, $n_{\text{latent}} = 2$, $n_{\text{layer}} = 64$, $\beta = 0$.

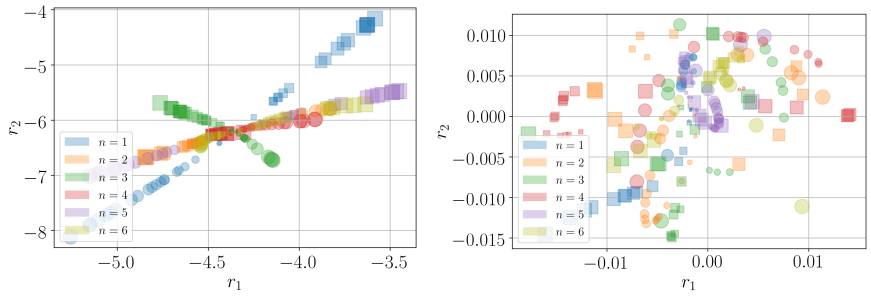

Figure 2: Values of the VAE's latent neurons, r_1 and r_2, for simulations in the datasets corresponding to configuration **a** (left) and **h** (right). Different colors represent different values of the wave number n. Circles represents positive values of the amplitude a, while squares represent negative values. In both cases, the marker size is proportional to $|a|$. We plot every fifth point in the respective datasets to improve legibility.

A further key question to investigate is whether the latent representations in the trained network are able to highlight the principal features of the map $f : (a, n) \to d_\star$, and might potentially be used to model this relationship in an accurate, explainable, and generalizable way (i.e., serve as the engine of an emulator in the sense of Approach I above). In particular, it is interesting to observe the role of the parameter β, as this is supposed to influence the distribution of simulations in the (r_1, r_2) space directly. In Figure 2, we plot the values of the network's latent neurons, r_1 and r_2, for all the simulations in two representative datasets: one from configuration **a** and one from configuration **h**.

We notice that, in the representation corresponding to $\beta = 0$, the models corresponding to different values of n split into different branches in the (r_1, r_2) plane, with a fairly regular dependence of both r_1 and r_2 on a. The representation also respects the basic property that the map between (a, n) and levee deformation ceases to be injective for $a = 0$, so that all the branches meet at one point. Interpolating (and, to an extent, extrapolating) this representation to obtain the value of the target observable for new (a, n) pairs should be straightforward. This relationship, combined with the network's decoder, would then act effectively as an emulator.

The representation obtained for $\beta = 0.8$ is much harder to interpret and, therefore, to model and use within an emulator architecture. We argue that minimizing statistical correlations might not be the right approach to ensure that the latent neurons represent independent degrees of freedom that control the physical model. How exactly this independence might be represented and enforced when training a VAE on data from physical simulations is an intriguing questions which we reserve for a further study.

REFERENCES

Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National*

Academy of Sciences, 113(15):3932–3937, 2016.

Agnimitro Chakrabarti, Steven R. Brandt, Qin Chen, and Fengyan Shi. Boussinesq modeling of wave-induced hydrodynamics in coastal wetlands. *Journal of Geophysical Research: Oceans*, 122(5):3861–3883, 2017.

Raban Iten, Tony Metger, Henrik Wilming, Lídia del Rio, and Renato Renner. Discovering Physical Concepts with Neural Networks. *Phys. Rev. Lett.*, 124(1):010508, January 2020. doi: 10.1103/PhysRevLett.124.010508.

Frank Loffler et al. The Einstein Toolkit: A Community Computational Infrastructure for Relativistic Astrophysics. *Class. Quant. Grav.*, 29:115001, 2012.

Hendrik Poulsen Nautrup, Tony Metger, Raban Iten, Sofiene Jerbi, Lea M. Trenkwalder, Henrik Wilming, Hans J. Briegel, and Renato Renner. Operationally meaningful representations of physical systems in neural networks, 2020.

Adam Oler, Ning Zhang, Steven R. Brandt, and Qin Chen. Implementation of an infinite-height levee in cafunwave using an immersed-boundary method. *Journal of Fluids Engineering*, 138 (11), 2/19/2021 2016. doi: 10.1115/1.4033490.

Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4), 2017.

Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009. ISSN 0036-8075. doi: 10.1126/science.1165893.

A WAVE DYNAMICS

In this section, we detail the quantitative aspects of the simulation model (Table 2), and show a random set of water-height plots for different levee geometries (Figures 3 and 4).

Table 2: Parameters of unperturbed model. All levels are measured with respect to the sea bottom.

Parameter	**Value**
Domain boundaries	$[0, 50m] \times [0, 50m]$
Spatial resolution	$0.2m$
x coordinate of the levee, $\bar{x}_L$	$25m$
y extension of the levee	$[10m, 40m]$
x interval of sea-land transition region	$[26m, 36m]$
Initial x location of wave peak	$15m$
Coastal elevation	$0.5m$
Initial peak height	$0.55m$
Physical time of evolution	$20s$

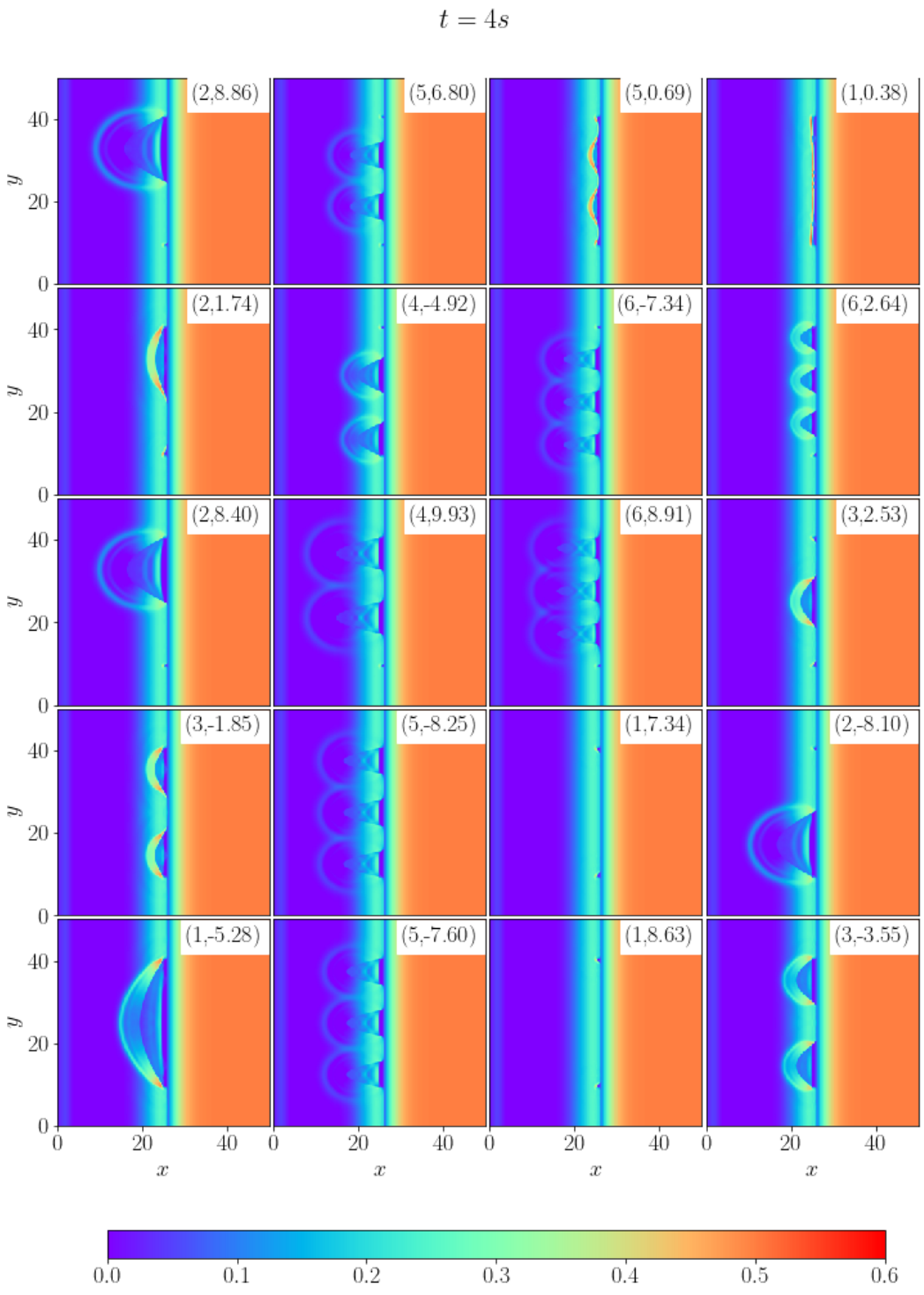

Figure 3: Random selection of simulation snapshots at $t = 4s$. Each plot is labelled by the corresponding (n, a) pair.

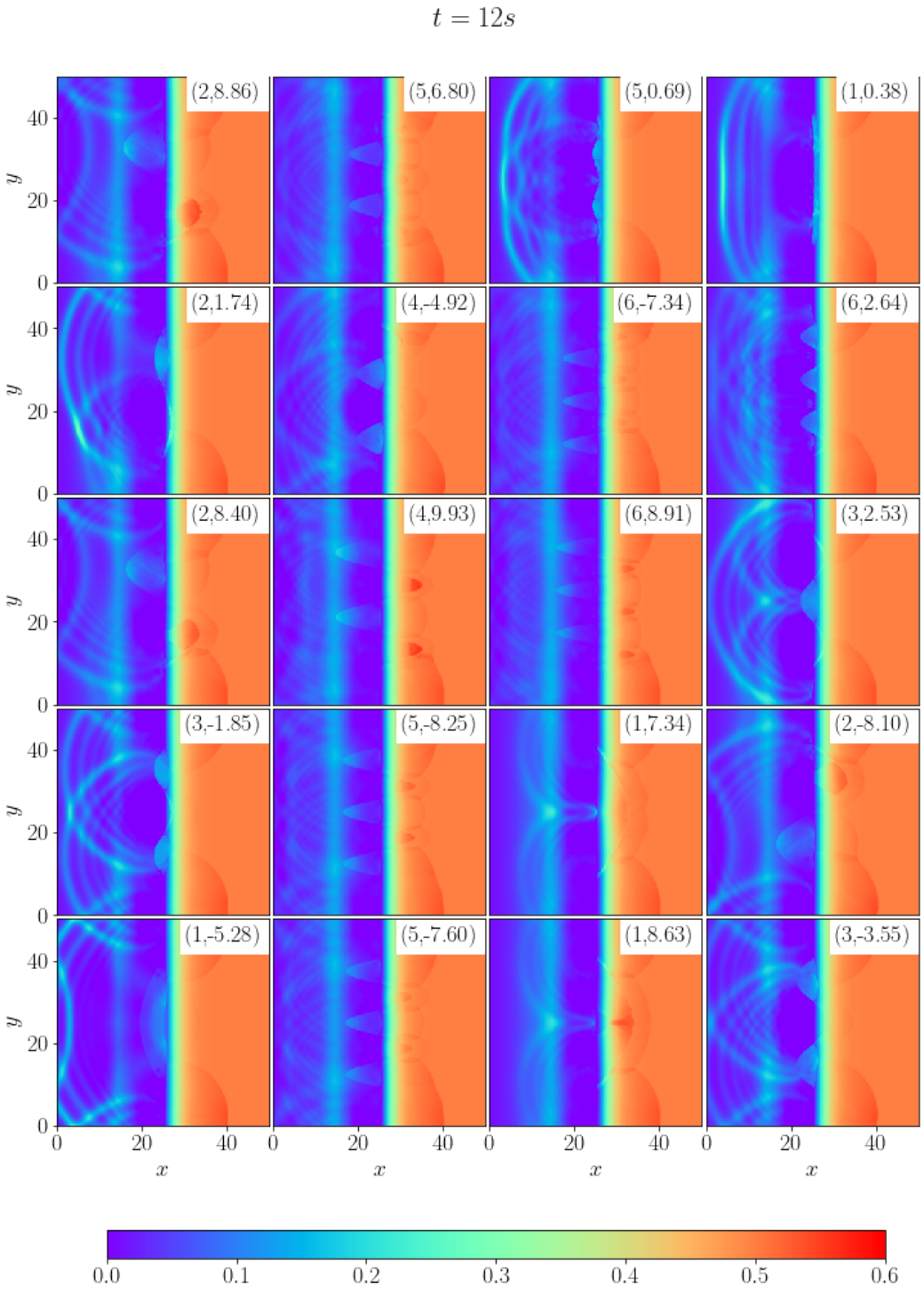

Figure 4: Same selection of simulations as in Figure 3, but at $t = 12s$.

B DATASETS AND RESULTS

In Figures 5-13, we show the complete results of the train and inference tasks corresponding to configurations **a** to **i**.

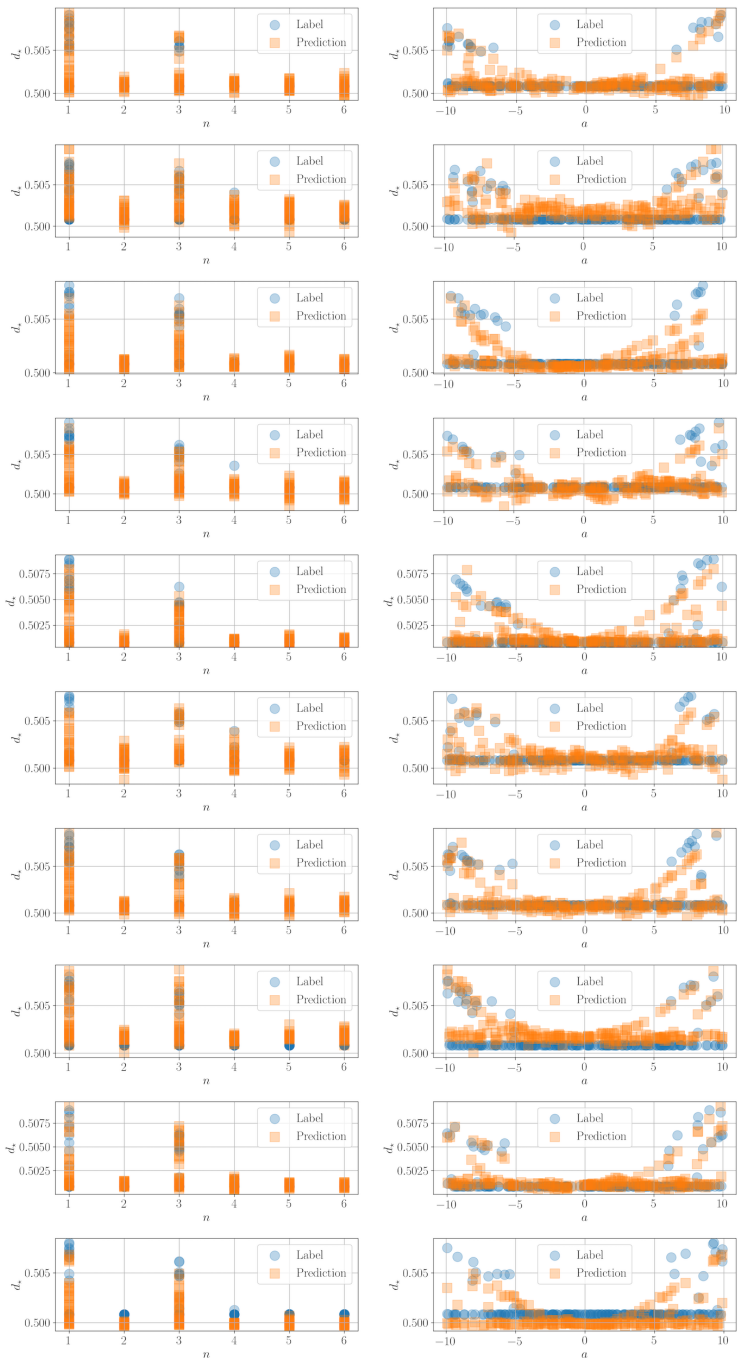

Figure 5: Inference results for $N_{\text{train}} = 10006$, $n_{\text{latent}} = 2$, $n_{\text{layer}} = 64$, $\beta = 0$ (configuration **a**).

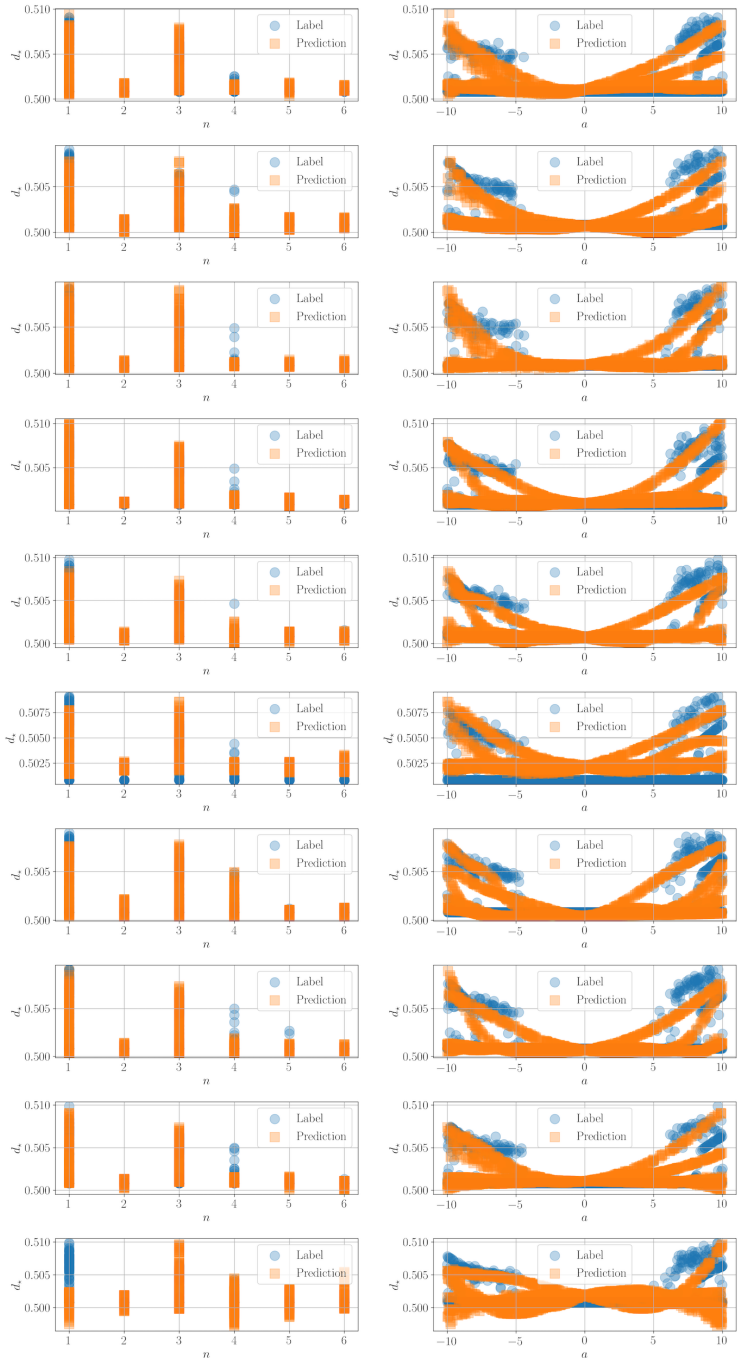

Figure 6: Inference results for $N_{\text{train}} = 10006$, $n_{\text{latent}} = 2$, $n_{\text{layer}} = 256$, $\beta = 0$ (configuration **b**).

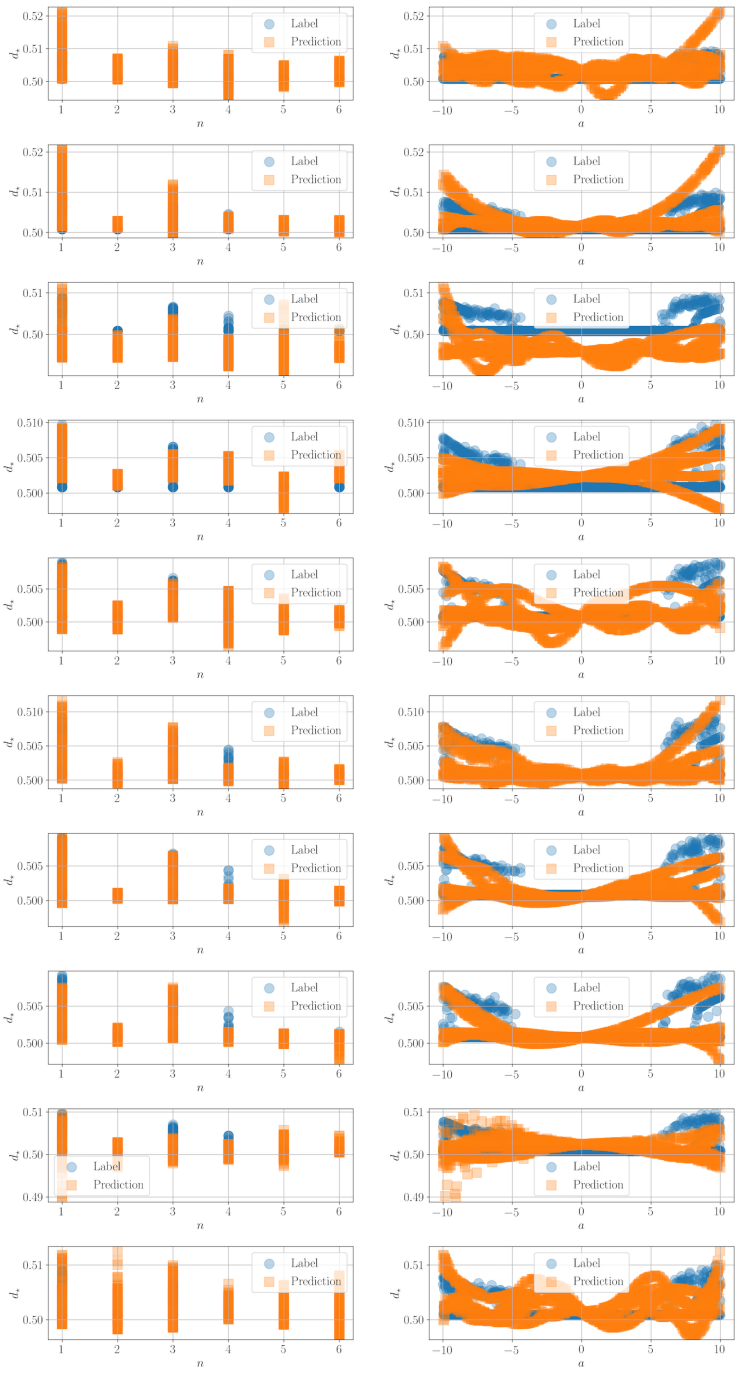

Figure 7: Inference results for $N_{\text{train}} = 10006$, $n_{\text{latent}} = 2$, $n_{\text{layer}} = 1024$, $\beta = 0$ (configuration **c**).

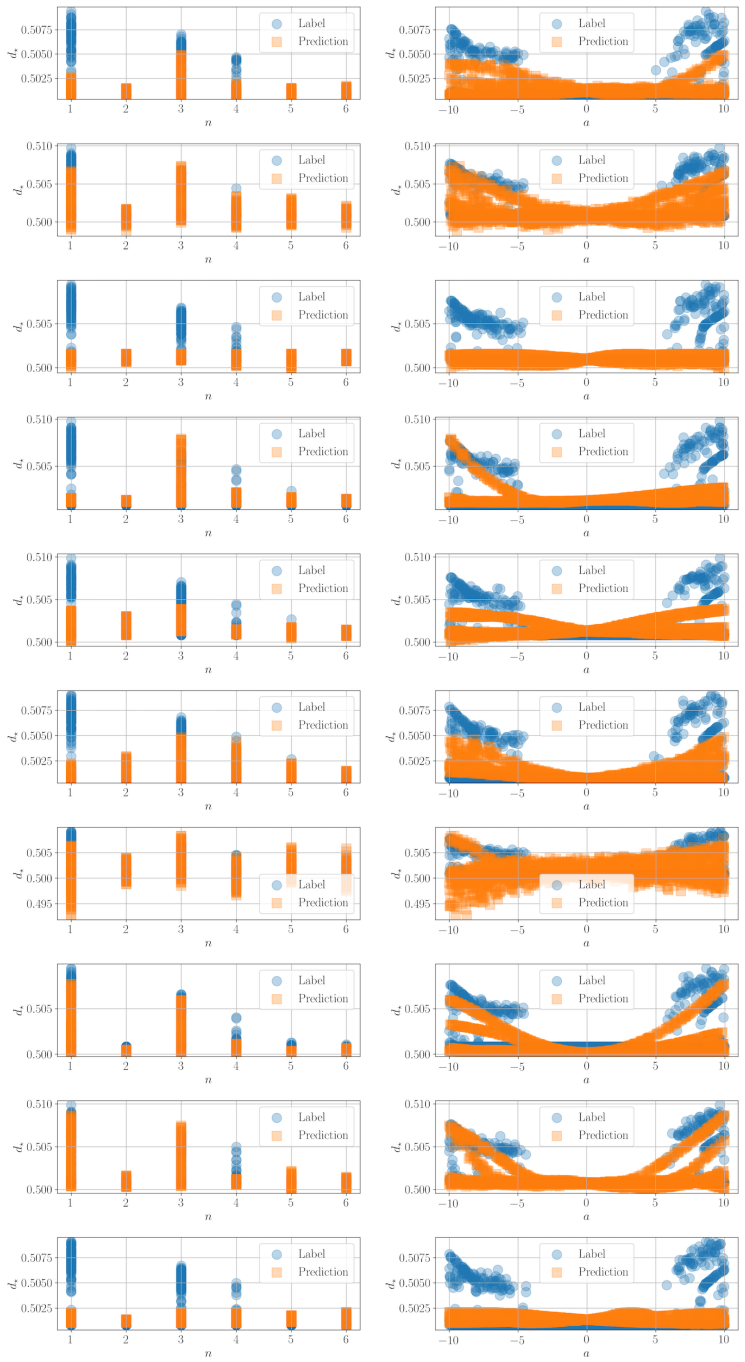

Figure 8: Inference results for $N_{\text{train}} = 10006$, $n_{\text{latent}} = 1$, $n_{\text{layer}} = 64$, $\beta = 0$ (configuration **d**).

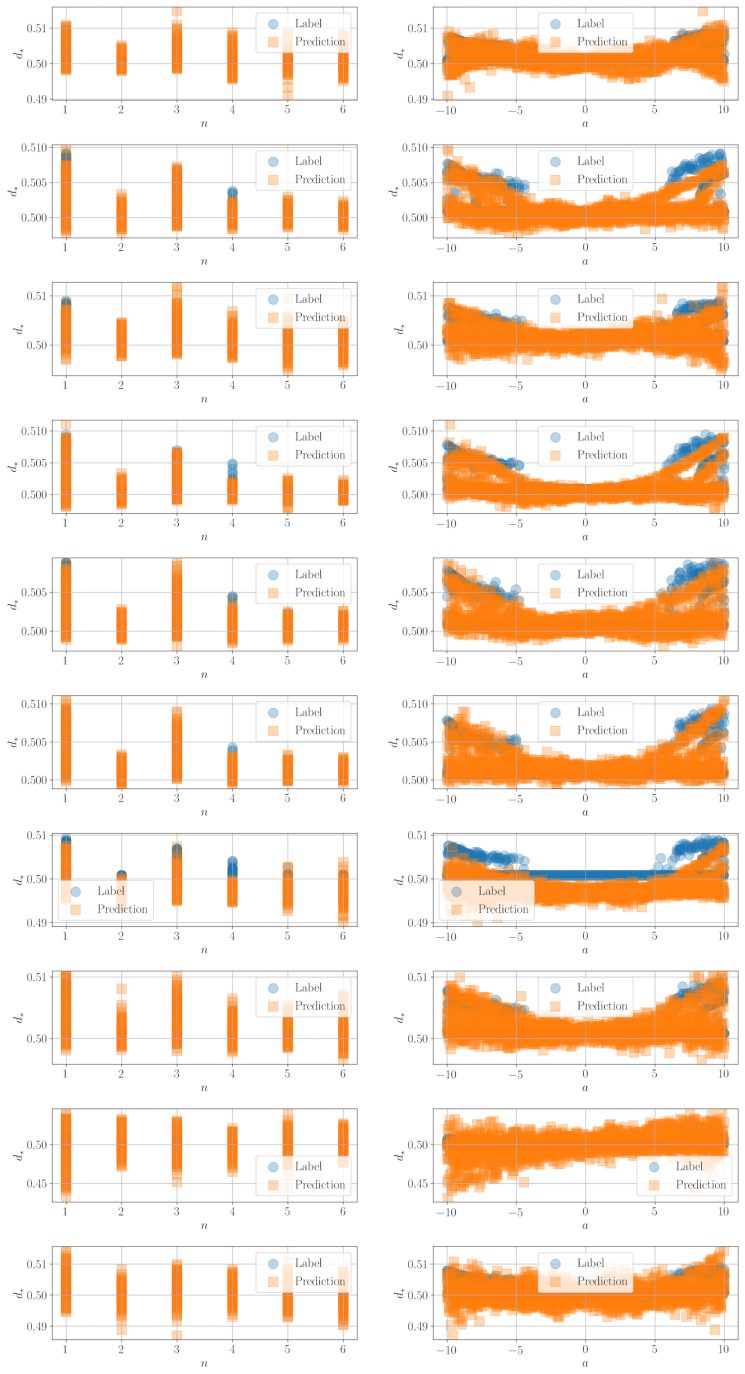

Figure 9: Inference results for $N_{\text{train}} = 10006$, $n_{\text{latent}} = 5$, $n_{\text{layer}} = 64$, $\beta = 0$ (configuration **e**).

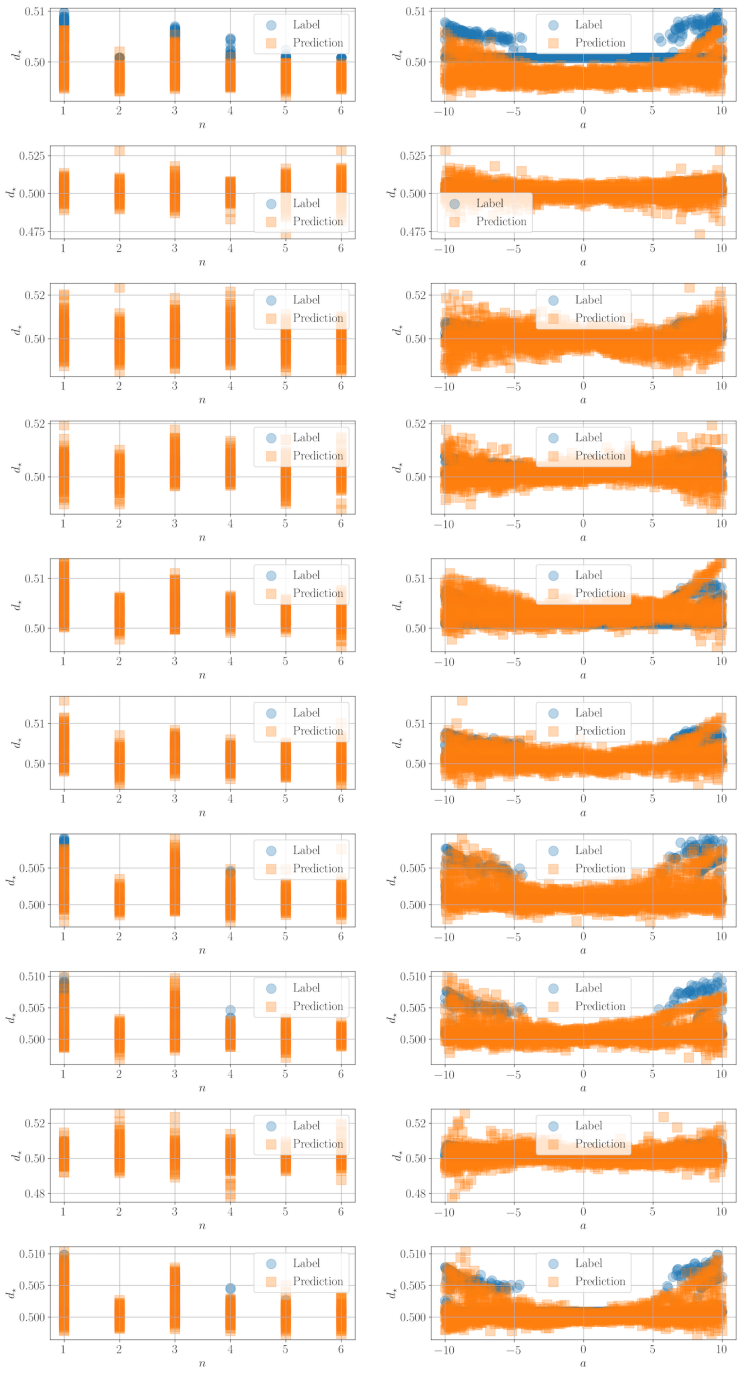

Figure 10: Inference results for $N_{\text{train}} = 10006$, $n_{\text{latent}} = 10$, $n_{\text{layer}} = 64$, $\beta = 0$ (configuration **f**).

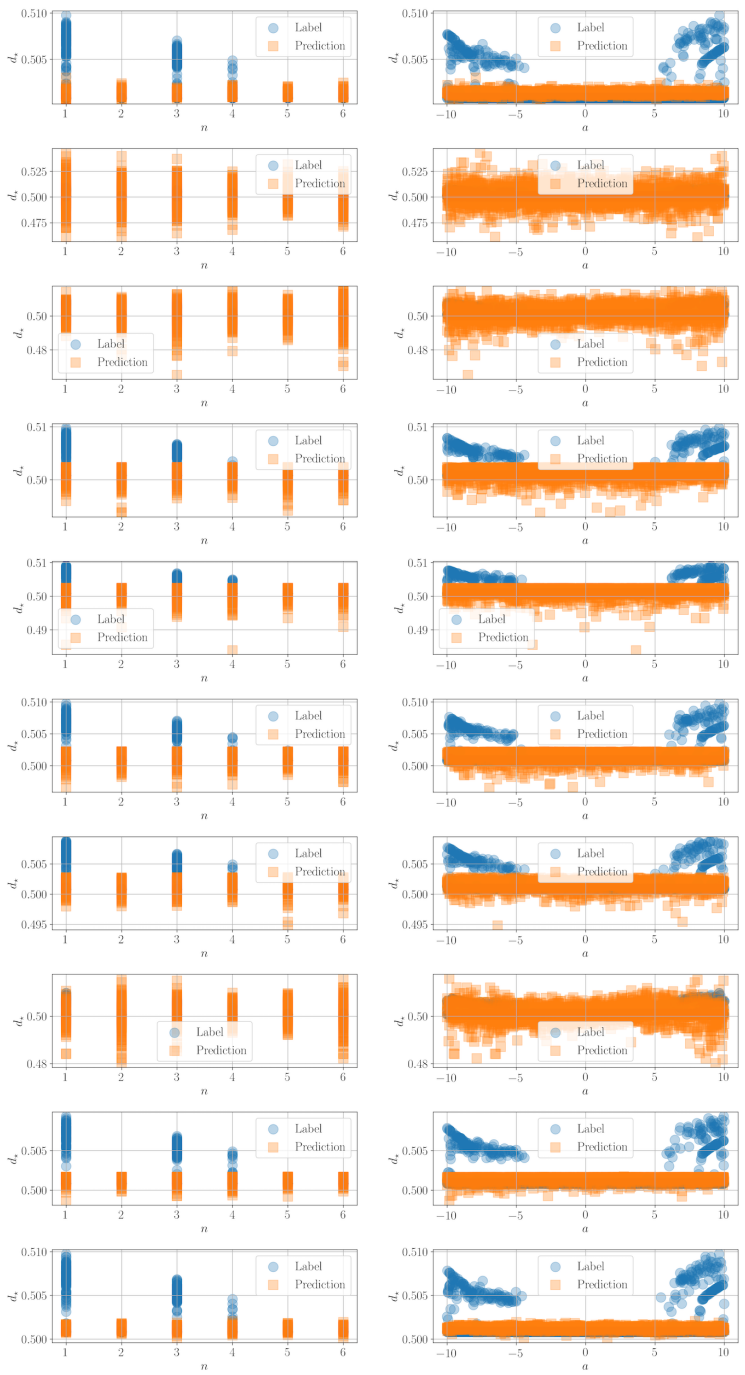

Figure 11: Inference results for $N_{\text{train}} = 10006$, $n_{\text{latent}} = 2$, $n_{\text{layer}} = 64$, $\beta = 0.2$ (configuration **g**).

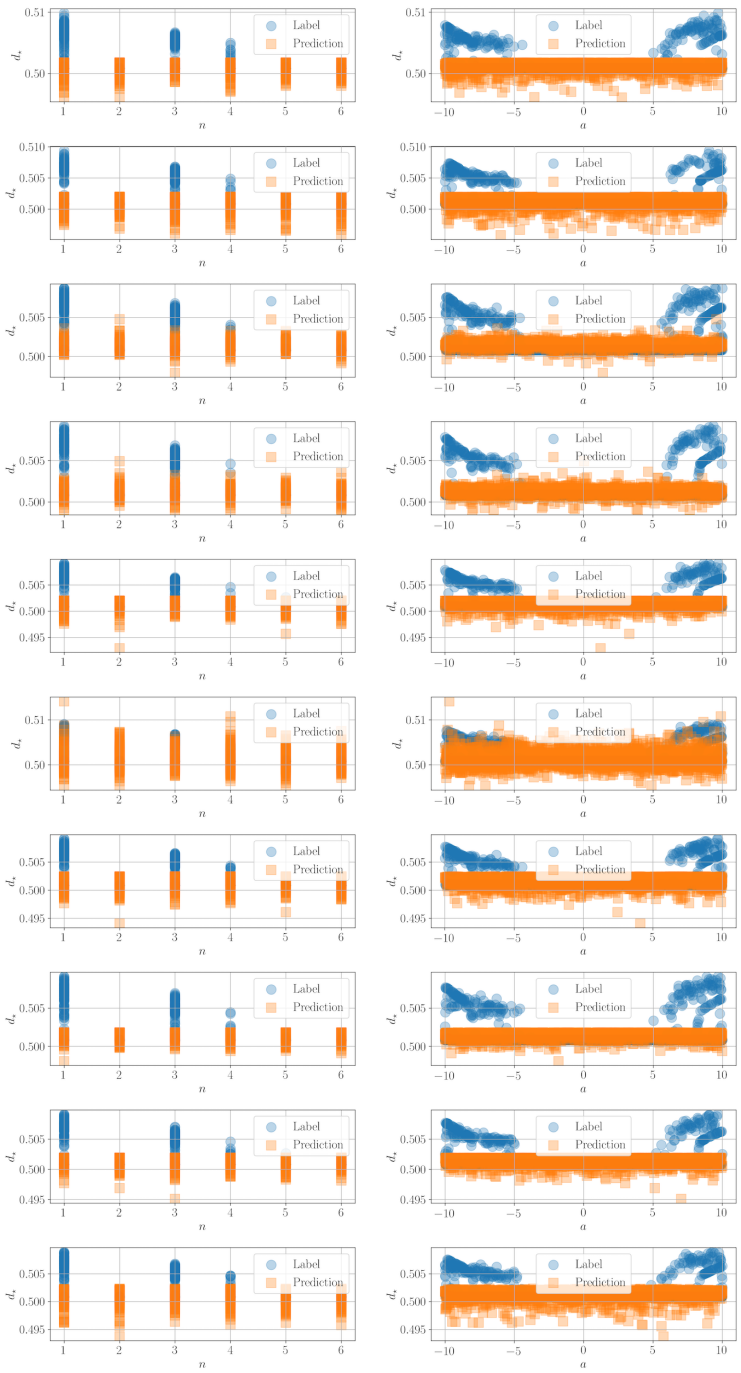

Figure 12: Inference results for $N_{\text{train}} = 10006$, $n_{\text{latent}} = 2$, $n_{\text{layer}} = 64$, $\beta = 0.8$ (configuration **h**).

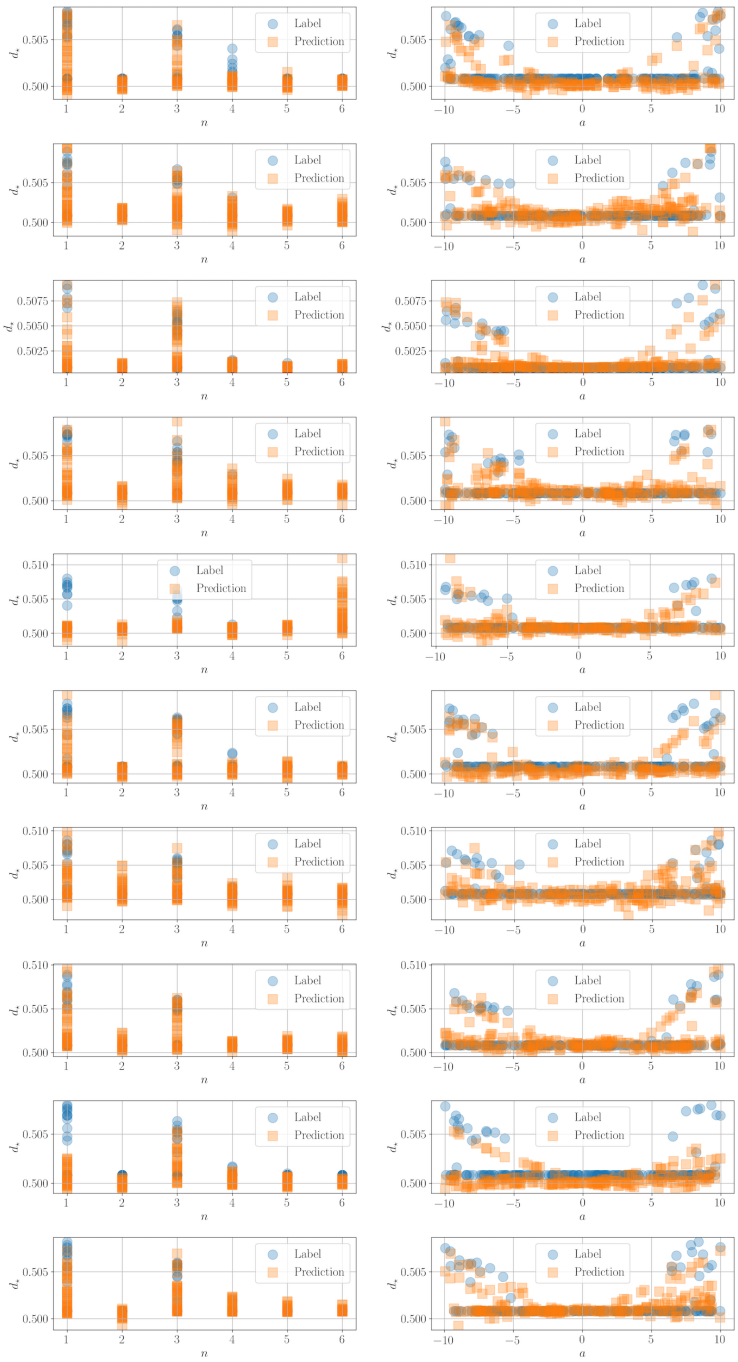

Figure 13: Inference results for $N_{\text{train}} = 11886$, $n_{\text{latent}} = 2$, $n_{\text{layer}} = 64$, $\beta = 0$ (configuration **i**).