2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Sparse neural networks with skip-connections for identification of aluminum electrolysis cell

Erlend Torje Berg Lundby, Haakon Robinson, Adil Rasheed, Ivar Johan Halvorsen, Jan Tommy Gravdahl

***Abstract*— Neural networks are rapidly gaining interest in nonlinear system identification due to the model's ability to capture complex input-output relations directly from data. However, despite the flexibility of the approach, there are still concerns about the safety of these models in this context, as well as the need for large amounts of potentially expensive data. Aluminum electrolysis is a highly nonlinear production process, and most of the data must be sampled manually, making the sampling process expensive and infrequent. In the case of infrequent measurements of state variables, the accuracy and open-loop stability of the long-term predictions become highly important. Standard neural networks struggle to provide stable long-term predictions with limited training data. In this work, we investigate the effect of combining concatenated skip-connections and the sparsity-promoting ℓ_1 regularization on the open-loop stability and accuracy of forecasts with short, medium, and long prediction horizons. The case study is conducted on a high-dimensional and nonlinear simulator representing an aluminum electrolysis cell's mass and energy balance. The proposed model structure contains concatenated skip connections from the input layer and all intermittent layers to the output layer, referred to as InputSkip. ℓ_1 regularized InputSkip is called sparse InputSkip. The results show that sparse InputSkip outperforms dense and sparse standard feedforward neural networks and dense InputSkip regarding open-loop stability and long-term predictive accuracy. The results are significant when models are trained on datasets of all sizes (small, medium, and large training sets) and for all prediction horizons (short, medium, and long prediction horizons.)**

I. Introduction

There is increasing interest in using machine learning-based methods to develop predictive models directly from data. Compared to standard system identification methods, the advantage of data-driven modeling is that it does not require any assumptions about the system. However, all phenomena well represented by the data can often be captured accurately. One example of such a method that has seen widespread popularity in recent years is the neural network (NN), which is known to be a universal function approximator. These are often used in Reinforcement Learning (RL) to represent a value function or a model for some dynamical system. However, this approach requires many data points to train effective models, which can be expensive in many domains. One hypothesis is that NNs are typically overparameterized and require many steps to adjust all parameters. However, overtraining on the same limited dataset will cause the model to overfit the training data and perform poorly on unseen data. While overparameterization has been found to aid convergence during training [1], it also introduces redundant information into the weights.

Recent research has found that sparser networks may be the key to training models that generalize across many situations. In particular, it has been shown empirically that for any dense architecture, there is a high probability that there is a sparse subnetwork that will train faster and generalize better than the full model [2]. This phenomenon is known as the Lottery Ticket Hypothesis. Many sparsification methods can be seen as attempts to extract such a "winning lottery ticket" from an initially dense network. In system identification, previous work shows that sparsity-promoting ℓ_1 regularization can benefit model generalization, interpretability, and stability [3]. Group sparsity methods have also been applied to Bayesian recurrent neural networks, with favorable results [4]. There have been numerous advances in this field, and we refer to [5] for a recent and comprehensive review. This work uses the well-known ℓ_1 regularization to induce sparsity in neural networks.

Another challenge related to using NNs is the choice of architecture and hyperparameters. Typical networks have multiple layers which are densely connected, although this can vary between domains. Choosing an appropriate architecture is an art involving trial and error to improve performance and avoid overfitting. It is commonly understood that the early layers of a neural network significantly impact the overall performance of the network. However, deep networks often suffer from the vanishing or exploding gradient problem, which prevents effective training of these early parameters [6]. Skip-connections were originally proposed to circumvent this by introducing a shorter path between the early layers and the output [7]. They were found to enable the training of significantly deeper networks but may also improve training convergence [8].

In the dynamical systems and control field, models are often designed with a purpose in mind, such as designing a control system or state observer. Crucially, we are interested in the behavior and performance of the controlled system regarding objectives such as energy efficiency or yield, implying that the model does not need to be perfectly accurate for the entire state space so long as the resulting closed-loop performance is sufficient (known as *identification for control* (I4C)). If high-frequency measurements from the system are available,

This work was supported by the project TAPI: Towards Autonomy in Process Industries (grant no. 294544), and EXAIGON: Explainable AI systems for gradual industry adoption (grant no. 304843)

E.T.B. Lundby, H. Robinson, A. Rasheed, and J. T. Gravdahl are with the Department of Engineering Cybernetics, Norwegian University of Science and Technology, 7034 Trondheim, Norway <erlend.t.b.lundby, haakon.robinson, adil.rasheed, jan.tommy.gravdahl>@ntnu.no

I. J. Halvorsen is with SINTEF Digital, Trondheim, No-7465, Norway ivar.j.halvorsen@sintef.no

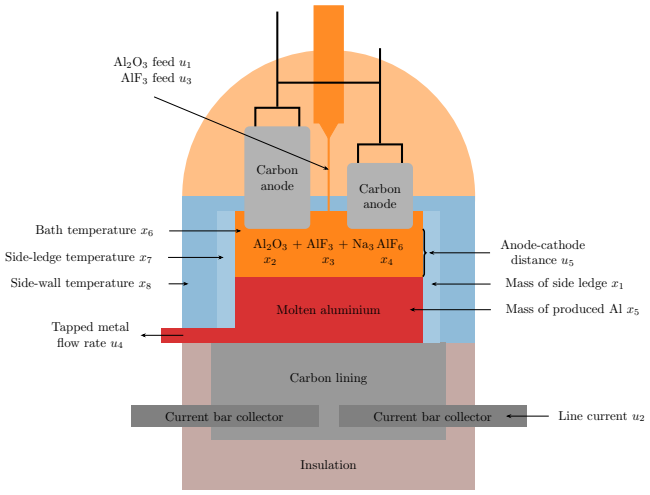


Fig. 1: Schematic of the setup

only the short-term behavior of the model is important since any drift out of the operational space is quickly corrected. However, if measurements are rarely available, such as in the aluminum electrolysis process that we consider, the long-term model behavior and open-loop stability become much more critical. Stable long-term predictions can be important for decision-making, meaning a model with good long-term stability and accuracy is inherently meaningful.

In this work, we investigate the effects of adding skip connections and ℓ_1 regularization on the accuracy and stability of these models for short, medium, and long horizons. The following questions are addressed:

- How do skip connections affect the stability and generalization error of neural networks trained on high-dimensional nonlinear dynamical systems?
- How does sparsity affect stability and generalization error for neural networks with skip connections when modeling nonlinear dynamics?
- How does the amount of training data affect neural networks with skip connections compared to neural networks without skip connections?

We make the following contributions:

- We perform a black box system identification of an aluminum electrolysis cell using different NN architectures.
- We demonstrate that the accuracy and open-loop stability of the resulting models is greatly improved by using ℓ_1 weight regularization and incorporating skip connections into the architecture.
- This advantage is consistent across datasets of varying sizes.

II. Theory

A. Physics-based model for aluminum extraction

NNs are first trained on synthetic data generated from a known physics-based models (PBM). The model used in this work describes the internal dynamics of an aluminum electrolysis cell based on the Hall-Héroult process. Fig. 1 shows a diagram of the electrolysis cell. Traditional PBMs of such systems are generally constructed by studying the mass/energy balance of the chemical reactions. The system is described by a set of ordinary differential equations (ODE):

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^8$ and $\mathbf{u} \in \mathbb{R}^5$ represent the time-varying states and inputs of the system respectively. The full set of equations are:

$$\dot{x}_1 = \frac{k_1(g_1 - x_7)}{x_1 k_0} - k_2(x_6 - g_1) \tag{2a}$$

$$\dot{x}_2 = u_1 - k_3 u_2 \tag{2b}$$

$$\dot{x}_3 = u_3 - k_4 u_1 \tag{2c}$$

$$\dot{x}_4 = -\frac{k_1(g_1 - x_7)}{x_1 k_0} + k_2(x_6 - g_1) + k_5 u_1 \tag{2d}$$

$$\dot{x}_5 = k_6 u_2 - u_4 \tag{2e}$$

$$\dot{x}_6 = \frac{\alpha}{x_2 + x_3 + x_4}\left[u_2 g_5 + \frac{u_2^2 u_5}{2620 g_2} - k_7(x_6 - g_1)^2 + k_8\frac{(x_6 - g_1)(g_1 - x_7)}{k_0 x_1} - k_9\frac{x_6 - x_7}{k_{10} + k_{11} k_0 x_1}\right] \tag{2f}$$

$$\dot{x}_7 = \frac{\beta}{x_1}\left[\frac{k_9(g_1 - x_7)}{k_{15} k_0 x_1} - k_{12}(x_6 - g_1)(g_1 - x_7) + \frac{k_{13}(g_1 - x_7)^2}{k_0 x_1} - \frac{x_7 - x_8}{k_{14} + k_{15} k_0 x_1}\right] \tag{2g}$$

$$\dot{x}_8 = k_{17} k_9\left(\frac{x_7 - x_8}{k_{14} + k_{15} k_0 \cdot x_1} - \frac{x_8 - k_{16}}{k_{14} + k_{18}}\right) \tag{2h}$$

where the intrinsic properties g_i of the bath mixture are given as:

$$g_1 = 991.2 + 112 c_{x_3} + 61 c_{x_3}^{1.5} - 3265.5 c_{x_3}^{2.2} - \frac{793 c_{x_2}}{-23 c_{x_2} c_{x_3} - 17 c_{x_3}^2 + 9.36 c_{x_3} + 1} \tag{3a}$$

$$g_2 = \exp\left(2.496 - \frac{2068.4}{273 + x_6} - 2.07 c_{x_2}\right) \tag{3b}$$

$$g_3 = 0.531 + 3.06 \cdot 10^{-18} u_1^3 - 2.51 \cdot 10^{-12} u_1^2 + 6.96 \cdot 10^{-7} u_1 - \frac{14.37(c_{x_2} - c_{x2,crit}) - 0.431}{735.3(c_{x_2} - c_{x2,crit}) + 1} \tag{3c}$$

$$g_4 = \frac{0.5517 + 3.8168 \cdot 10^{-6} u_2}{1 + 8.271 \cdot 10^{-6} u_2} \tag{3d}$$

$$g_5 = \frac{3.8168 \cdot 10^{-6} g_3 g_4 u_2}{g_2(1 - g_3)}. \tag{3e}$$

See Table I for a description of these quantities. The dynamics of the system are relatively slow. The control inputs u_1, u_3 and u_4 are therefore well modeled as impulses representing discrete events involving the addition or removal of substances. This results in step changes in the linear states x_2, x_3, x_5, which act as accumulator states for the mass of the corresponding substance (see Table I). The control inputs u_2 and u_5 are piecewise constant and nonzero. The inputs $\mathbf{u}$ are determined by a simple proportional controller $\boldsymbol{\pi}(\mathbf{x})$. The simulation model is derived in [3], and we refer to that

TABLE I: Table of states, inputs, and other quantities used to model the electrolysis cell

Variable	Physical meaning	Units
x_1	Mass side ledge	kg
x_2	Mass Al_2O_3	kg
x_3	Mass AlF_3	kg
x_4	Mass $Na_3\,AlF_6$	kg
x_5	Mass metal	kg
x_6	Temperature bath	°C
x_7	Temperature side ledge	°C
x_8	Temperature side wall	°C
u_1	Al_2O_3 feed	kg/s
u_2	Line current	kA
u_3	AlF_3 feed	kg/s
u_4	Aluminum tapping	kg/s
u_5	Anode-cathode distance	cm
c_{x_2}	Al_2O_3 mass ratio $x_2/(x_2+x_3+x_4)$	-
c_{x_3}	AlF_3 mass ratio $x_3/(x_2+x_3+x_4)$	-
g_1	Liquidus temperature	°C
g_2	Electrical conductivity	S m
g_3	Bubble coverage	-
g_4	Bubble thickness	cm
g_5	Bubble voltage	V

article for the values of the simulation parameters and further details.

B. Deep neural network with skip connections

A NN with L layers can be compactly written as an alternating composition of affine transformations $\mathbf{W}\mathbf{z}+\mathbf{b}$ and nonlinear activation functions $\boldsymbol{\sigma}:\mathbb{R}^n\mapsto\mathbb{R}^n$:

$$\mathbf{z}_i=\boldsymbol{\sigma}_i\left(\mathbf{W}_i\mathbf{z}_{i-1}+\mathbf{b}_i\right) \tag{4}$$

where $\mathbf{z}_0$ is the input to the network, the activation function $\boldsymbol{\sigma}_i$, weight matrix $\mathbf{W}_i$, and bias vector $\mathbf{b}_i$ correspond to the ith layer of the network. The universal approximation property of NNs makes them very attractive as a flexible model class when a lot of data is available. The representation capacity is generally understood to increase with both the depth and the width (the number of neurons in each layer), although early attempts to train very deep networks found them challenging to optimize using backpropagation due to the vanishing gradients problem. One of the major developments that enabled researchers to train deep NNs with many layers is the *skip connection*. A skip connection is simply an additional inter-layer connection that bypasses some of the layers of the network. This provides alternate pathways through which the loss can be backpropagated to the early layers of the NN, which helps mitigate the issues of vanishing and exploding gradients, which were major hurdles to training deeper models. In this work, we utilize a modified DenseNet architecture as proposed by [9], where the outputs of earlier layers are concatenated to all the consecutive layers. We simplify the structure such that the model only contains skip connections from the input layer to all consecutive layers. We call this architecture InputSkip, which has reduced complexity

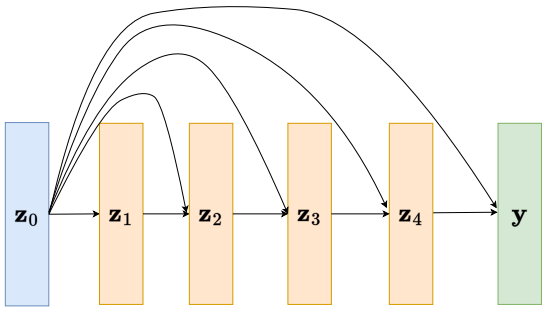


Fig. 2: InputSkip architecture with 4 hidden layers

compared to DenseNet:

$$\begin{aligned}\mathbf{z}_1&=\boldsymbol{\sigma}_1(\mathbf{W}_1\mathbf{z}_0+\mathbf{b}_1)\\ \mathbf{z}_i&=\boldsymbol{\sigma}_i\left(\mathbf{W}_i\begin{bmatrix}\mathbf{z}_{i-1}\\ \mathbf{z}_0\end{bmatrix}+\mathbf{b}_i\right),i>1\end{aligned} \tag{5}$$

The output of each layer (excl. the first) becomes a sum of a linear and a nonlinear transformation of the initial input $\mathbf{x}$. Hence, the skip connections from the input layer to consecutive layers facilitate the reuse of the input features for modeling different linear and nonlinear relationships more independently. The InputSkip architecture with four hidden layers is illustrated in Figure 2.

III. Method and Setup

In this section, we present all the details of data generation, its preprocessing, and the methods required to reproduce the work. The steps can be briefly summarized as follows:

- Simulate (2) with RK4 scheme and random initial conditions to generate 140 trajectories with 5000 timesteps each.
- Set aside 40 for training and 100 for testing. Construct three datasets by selecting 10,20, and 40 trajectories, respectively.
- For each model class and dataset, train ten instances on the training data.
- Repeat all experiments with ℓ_1 regularization, see loss function in (7).
- Use trained models to generate predicted trajectories along the test set and compare them to the 100 test trajectories.
- All models are implemented using the `PyTorch` library [10].

A. Data generation

The state trajectories used in the test and training sets were generated by integrating Equation (2) with the numerical RK4 integration scheme with a fixed timestep $h=10\,\mathrm{s}$ on the interval $[0,5000h]$. This was found to be sufficient for the relatively slow dynamics of the system. The initial conditions were sampled uniformly from the intervals shown in Table II to generate 140 unique trajectories. A total of 40 trajectories were set aside for training and 100 of the trajectories as a test set. The 40 training trajectories were used to create three datasets of varying sizes (small, medium, large), namely

TABLE II: Initial conditions intervals for $\mathbf{x}$

Variable	Initial condition interval
x_1	[2060, 4460]
c_{x_2}	[0.02, 0.05]
c_{x_3}	[0.09, 0.13]
x_4	[11500, 16000]
x_5	[9550, 10600]
x_6	[940, 990]
x_7	[790, 850]
x_8	[555, 610]

10, 20, and 40 trajectories, containing 50000, 100000, and 200000 individual data points.

Equation (2) also depends on the input signal $\mathbf{u}$. In practice, this is given by a deterministic control policy $\mathbf{u} = \boldsymbol{\pi}(\mathbf{x})$ that stabilizes the system and keeps the state $\mathbf{x}$ within some region of the state space that is suitable for safe operation. We found that this was insufficient to successfully train our models because the controlled trajectories showed minimal variation after some time, despite having different initial conditions. This lack of diversity in the dataset resulted in models that could not generalize to unseen states, which frequently arose during evaluation. To inject more variety into the data and sample states $\mathbf{x}$ outside of the standard operational area, we used a stochastic controller

$$\boldsymbol{\pi}_s(\mathbf{x}) = \boldsymbol{\pi}(\mathbf{x}) + \mathbf{r}(t)$$

that introduced random perturbations $\mathbf{r}(t)$ to the input. These perturbations were sampled using the Amplitude-modulated Pseudo-Random Binary Signal (APRBS) method proposed by [11].

In system identification, it is typical to optimize the model to estimate the function $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$. However, this is not feasible for (2) because the inputs $\mathbf{u}$ are not differentiable. Instead, the trajectories are discretized using the forward Euler difference:

$$\mathbf{y}_k = \frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{h} \tag{6}$$

The datasets are then constructed as sets of the pairs $([\mathbf{x}_k, \mathbf{u}_k], \mathbf{y}_k)$. In practice, measurements will be noisy and the state trajectories must be estimated using a filtering method, e.g., moving horizon estimation.

B. Model architectures

Two different architectures are evaluated in this case study: a standard feed-forward Multi-Layer Perceptrons (MLP) referred to as PlainNet and the modified MLP with concatenated skip-connections from the input layer called InputSkip, see Fig. 2 for illustration. Moreover, both structures are trained with and without the sparsity promoting ℓ_1 regularization (as done in [3]), yielding four model structures: PlainDense, PlainSparse, InputSkipDense, and InputSkipSparse. The input layer of each of the models is the concatenation of the measured state $\mathbf{x}_k \in \mathbb{R}^8$, or the estimated state $\hat{\mathbf{x}}_k \in \mathbb{R}^8$ at timestep k, and the control input vector $\mathbf{u}_k \in \mathbb{R}^8$ at timestep k, yielding a vector $\mathbf{z}_0 = \{\mathbf{x}_k,\ \mathbf{u}_k\} \in \mathbb{R}^{13}$. Each of the structures has four hidden layers, and each of the layers has 25 neurons. In addition, the input vector $\mathbf{z_0}$ is concatenated to each of the hidden layers in the InputSkip structures, such that each of the hidden layers in the InputSkip structures has $25 + 13 = 38$ states. All models output an estimate of the time derivative of the state variables $\dot{\mathbf{x}}_k \in \mathbb{R}^8$ at timestep k. Each model class's sparse and dense structures start with the same architecture before training, but for sparse structures, many of the neurons and weights are zeroed out by the ℓ_1 regularization term. Since InputSkip has more states in the hidden layers than Plain structures due to the input vector being concatenated to each layer's output, it is reasonable to ask whether the Plain structures should have more neurons in the hidden layers. This is tested, and it turns out that this does not benefit the structures regarding the evaluation measures.

C. Training setup

The models are trained by minimizing the following loss function using stochastic gradient descent:

$$\mathbf{J}_\theta = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} (\mathbf{y_i} - \hat{\mathbf{f}}(\mathbf{x_i}, \mathbf{u_i}))^2 + \lambda \sum_{j=1}^{L} |\mathbf{W}_j| \tag{7}$$

where the *batch* $\mathcal{B}$ is a set of indices corresponding to a random subset of examples from the data, L is the number of layers of the NN, and λ is the regularization parameter. This loss function is the sum of the mean squared error (MSE) of the model $\hat{\mathbf{f}}$ with respect to the regression variables $\mathbf{y}$, and the ℓ_1 norm of the connection weight matrices $\mathbf{W}_i$ in all layers. We used a batch size of $|\mathcal{B}| = 128$. We used the popular ADAM solver proposed by [12] with default parameters to minimize (7). The dense model structures PlainDense and InputSkipDense were trained without ℓ_1 regularization (i.e. $\lambda = 0$), and the sparse model structures PlainSparse and InputSkipSparse were trained with $\lambda = 10^{-4}$.

D. Evaluation of model accuracy

Starting from a given initial condition $\mathbf{x}(t_0)$, the model $\hat{\mathbf{f}}(\mathbf{x}, \mathbf{u})$ is used to generate an estimated trajectory using the recurrence:

$$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + h\, \hat{\mathbf{f}}(\hat{\mathbf{x}}_k, \mathbf{u}_k) \tag{8}$$

where $\hat{\mathbf{x}}_0 = \mathbf{x}_0$. Applying multi-step or higher order Runge-Kutta methods to NN models is possible. However, these methods require multiple evaluations of the model per timestep, which increases the computation and memory needed to perform automatic differentiation. The forward Euler method is preferred, as it only evaluates the model once per timestep. In the approach outlined here, this does not incur significant discretization errors, as (8) effectively reverses the discretization step in (6). The input signal $\mathbf{u}_k$ is sampled directly from the test trajectory. Borrowing a term from the field of time-series analysis, this is referred to as a *rolling forecast*. To evaluate the accuracy of a model over multiple trajectories, we define the Average Normalized Rolling Forecast Mean Squared Error (AN-RFMSE):

$$\text{AN-RFMSE} = \frac{1}{p} \sum_{i=1}^{p} \frac{1}{n} \sum_{j=1}^{n} \left(\frac{\hat{x}_i(t_j) - x_i(t_j)}{\text{std}(x_i)} \right)^2 \tag{9}$$

where $\hat{x}_i(t_j)$ is the model estimate of the simulated state variable x_i at time step t_j, $\mathrm{std}(x_i)$ is the standard deviation of variable x_i in the training set $\mathcal{S}_{train}$, $p = 8$ is the number of state variables and n is the number of time steps being averaged over.

E. Evaluation of model stability

A symptom of model instability is that its predictions can *blow up*, characterized by a rapid (often exponential) increase in prediction error. More precisely, a blow-up is said to occur when all system states' normalized mean absolute error exceeds three (this corresponds to standard deviations):

$$\max_{j<n} \left[\frac{1}{p} \sum_{i=1}^{p} \left(\frac{|\hat{x}_i(t_j) - x_i(t_j)|}{\mathrm{std}(x_i)} \right) \right] > 3 \qquad (10)$$

where $p = 8$ is again the number of state variables and n is the number of time steps to consider. Equation (10) is conservative. However, this does not lead to a significant underestimation of the number of blow-ups. This is because once a model starts to drift rapidly, it quickly exceeds the threshold.

IV. Results and discussions

This section reports empirical results for the model accuracy and stability of the different model classes (PlainDense, PlainSparse, InputSkipDense, InputSkipSparse). A Monte Carlo analysis is performed by training ten instances of each model class and evaluating these on the test set consisting of 100 trajectories, where each trajectory has a length of 5000 timesteps. The test set is generated as described in Section III-A, using the simulation model in Equation (2). The evaluation procedure follows; the models are given initial values for each state variable trajectory in the test set. Then, the models forecast state values at the consecutive time steps for each test set trajectory as described in (8) without feedback from measurements of the state variables. The resulting forecasts are evaluated according to the prediction accuracy measure in (9) and the forecast stability measure in (10). The prediction accuracy of different model classes is reported in Fig. 4, and the empirical model stability results are reported in Fig. 3. Accuracy and stability are reported for different forecasting horizons. We repeat the experiments for all model classes trained on three different dataset sizes to study the data efficiency of the models.

Fig. 3 presents the total number of blow-ups recorded within each model class after $100h$, $2500h$, and $5000h$ (short, medium, and long term respectively). For simplicity, blow-ups were detected by thresholding the computed variance of a predicted trajectory. It is clear that for short time horizons, all the models exhibit robust behavior independently of the size of the training datasets. However, for medium and long time horizons, PlainDense, PlainSparse, and InputSkipDense architectures exhibit a significant number of blow-ups and, therefore, instability. Figs. 3a - 3c show that PlainDense is generally the most unstable, with up to 41% of all trajectories resulting in a blow-up. For the smallest amount of training data (see Fig. 3a) PlainSparse and InputSkipDense have similar blow-up frequencies. The PlainSparse architecture shows significantly better stability for larger datasets than both PlainDense and InputSkipDense. The PlainDense and PlainSparse models blow up slightly less often when trained on the medium-sized dataset rather than the largest dataset. In contrast, the blow-up rate InputSkipDense models decreases with increased data availability. However, all three of these models still suffer from high blow-up rates.

In comparison, almost no blow-ups are recorded using the InputSkipSparse architecture, even for the small training dataset. In Figure 3, the orange bars corresponding to the blow-up frequency of InputSkipSparse models are not visible for any training sets due to the significantly lower number of blow-ups. For InputSkipSparse models trained on the smallest dataset, only 3 out of 1000 possible blow-ups were reported for the longest horizon. Apart from that, no blow-ups were reported for the InputSkipSparse models.

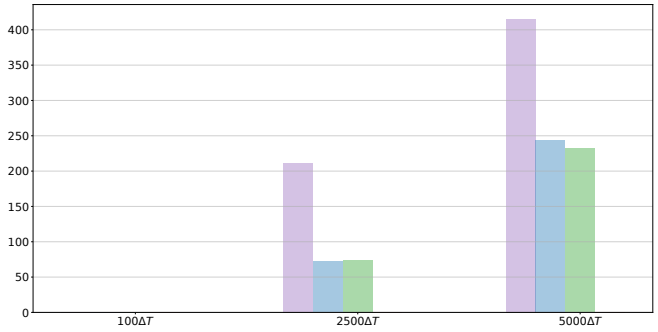
Fig. 4 presents a violin plot of the accuracy of each model class, expressed in terms of AN-RFMSE over different time horizons. A larger width of the violin indicates a higher density of that given RFMSE value, while the error bars show the minimum and maximum recorded RFMSE values. The model estimates that blew up (see Fig. 3) are excluded as outliers. In this way, the generalization performance of the models is estimated only within their regions of stability. A potential pitfall of excluding these outliers is that model classes that blow up often have their worst scores removed, thus biasing the distribution towards lower scores. Despite this, low accuracy appears to correlate with a high blow-up rate. The InputSkipSparse architecture is consistently more accurate (up to an order of magnitude) than the others in the long term.

Fig. 5 shows the effect of InputSkipSparse compared to PlainSparse for a single representative test-set trajectory and is not meant to the significance of the results. The significance of the results can be found in Fig. 4 and Fig. 3, which show results for the entire test set.

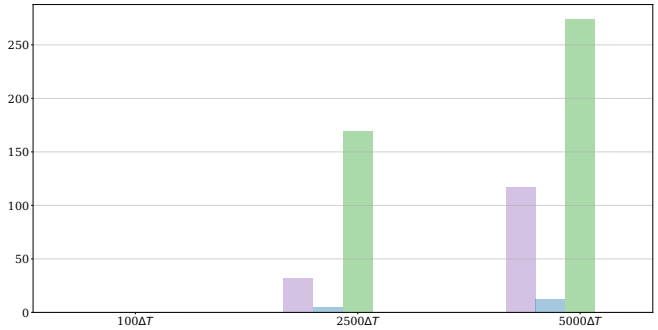
V. Conclusion and future work

This work compared the performance of two different model structures trained with and without sparsity promoting ℓ_1 regularization. The two model types are standard MLP and a more specialized architecture that includes skip connections from the input layer to all consecutive layers, yielding four different model structures: PlainDense, PlainSparse, InputSkipDense, and InputSkipSparse. The main conclusions of the article are as follows:

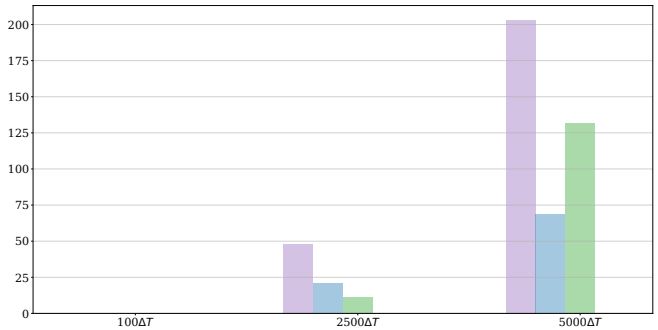
- NNs with skip connections are more stable for predictions over long time horizons compared to standard MLPs. Furthermore, the accuracy of NNs with skip connections is consistently higher for all forecasting horizons.
- The application of sparsity-promoting ℓ_1 regularization significantly improves the stability of the standard MLP and InputSkip architectures. This improvement was more apparent for models with the InputSkip architecture.



(a) Trained on smallest dataset with 50000 datapoints



(b) Trained on medium sized dataset with 100000 datapoints

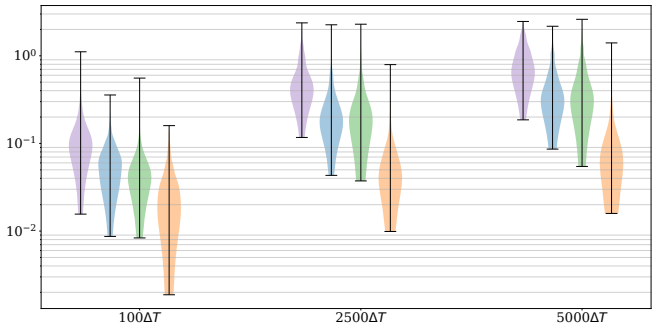


(c) Trained on largest dataset with 200000 datapoints

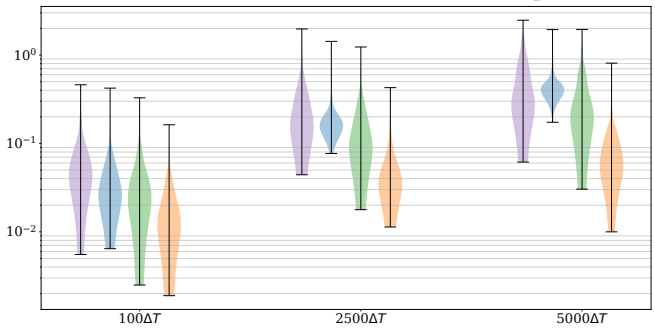
Fig. 3: Divergence plot: Number of trajectories that blow-up over different time horizons according to the measure defined in Equation (10). The total number of trajectories is 1000, so the values can be read as a permille. The InputSkipSparse model only diverged 3 times (small dataset at $5000\Delta T$), hence the absence of the corresponding bar.

- The InputSkipSparse showed satisfactory stability characteristics even when the amount of training data was restricted, suggesting that this architecture is more suitable for system identification tasks than the standard MLP structure.

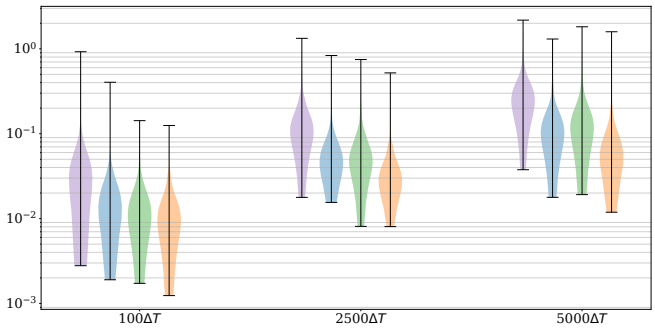
The case study shows that both sparsity-promoting regularization and skip connections can result in more stable NN models for system identification tasks while requiring fewer data and improving their multi-step generalization for both short, medium, and long prediction horizons. Despite the encouraging performance of the sparse-skip networks, it is yet to be determined if the benefits also extend to the case



(a) Trained on smallest dataset with 50000 datapoints

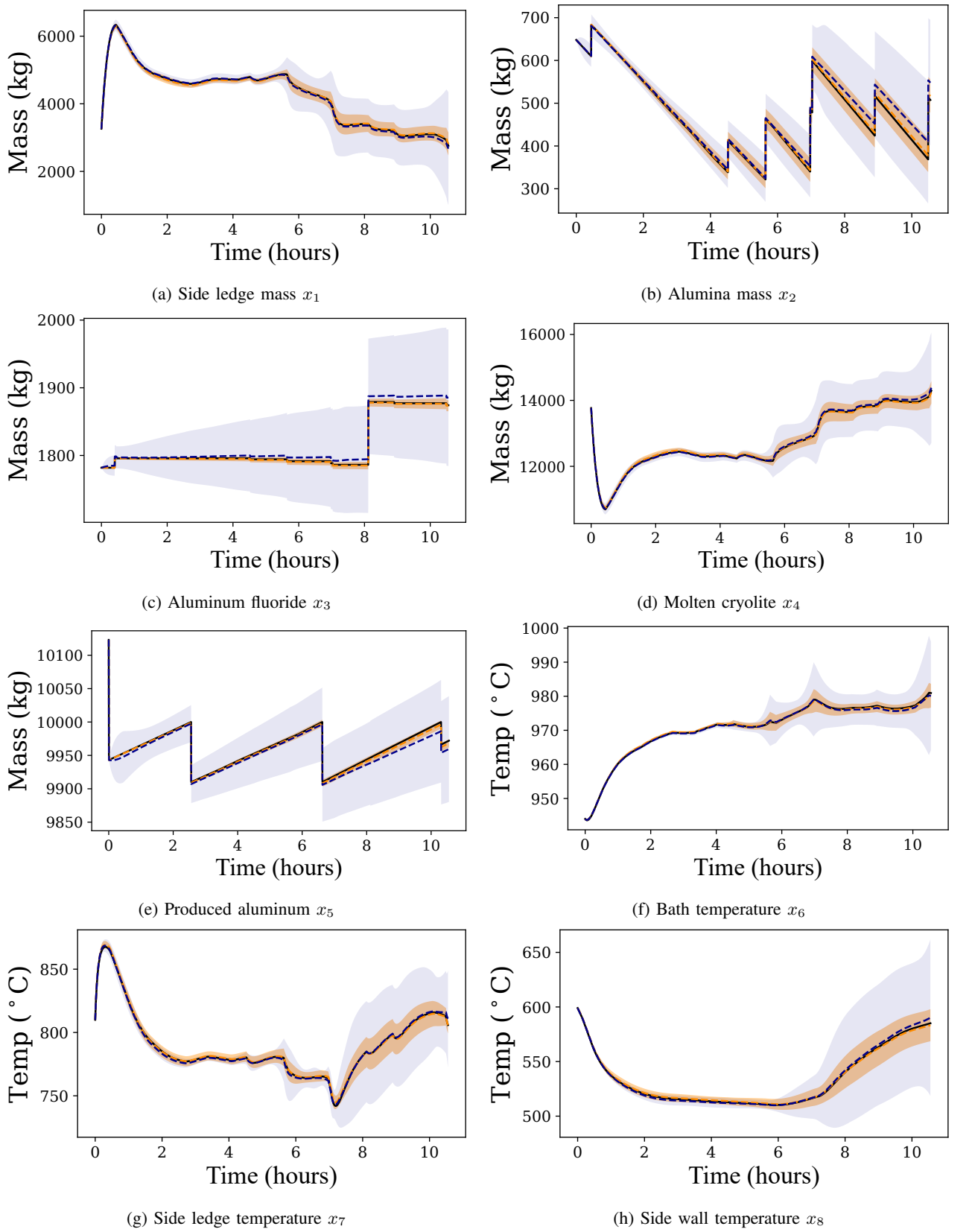


(b) Trained on medium sized dataset with 100000 datapoints



(c) Trained on largest dataset with 200000 datapoints

Fig. 4: Model accuracy is expressed in terms of AN-RFMSE over different horizons. AN-RFMSE is defined for a single model forecast of a single trajectory in (9). Ten models of each of the model types (PlainDense, PlainSparse, InputSkipDense, InputSkipSparse) are trained on the 50000 data points in Figure 4a, 100000 data points in Figure 4b, and 200000 data points in Figure 4c. The model estimates that blow up (see Figure 3) are excluded. The plot shows that sparse models with skip connections (InputSkipSparse) are consistently more accurate than sparse and dense models without skip connections.



(a) Side ledge mass x_1

(b) Alumina mass x_2

(c) Aluminum fluoride x_3

(d) Molten cryolite x_4

(e) Produced aluminum x_5

(f) Bath temperature x_6

(g) Side ledge temperature x_7

(h) Side wall temperature x_8

Fig. 5: Rolling forecast of a representative trajectory from the test set (100 trajectories total)

of noisy measurements.

This case study also has relevance beyond the current case study. In more realistic situations, we often have a partial understanding of the system we wish to model (see (2)) and only wish to use data-driven methods to correct a PBM when it disagrees with the observations (e.g., due to a faulty assumption). As shown in [13], combining PBMs and data-driven methods in this way also has the potential to inject instability into the system. Another concern is the safety of the real world system when collecting training data. In more practical settings, techniques such as the APRBS method should be combined with constraint satisfaction mechanisms (e.g. predictive safety filters [14]). Finding new ways to improve or guarantee out-of-sample behavior for data-driven methods is therefore paramount to improving such systems' safety.

REFERENCES

[1] Z. Allen-Zhu, Y. Li, and Z. Song, "A convergence theory for deep learning via over-parameterization," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 242–252, PMLR, 09–15 Jun 2019.

[2] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *International Conference on Learning Representations*, 2019.

[3] E. T. B. Lundby, A. Rasheed, J. T. Gravdahl, and I. J. Halvorsen, "Sparse deep neural networks for modeling aluminum electrolysis dynamics," *Applied Soft Computing*, vol. 134, p. 109989, Feb. 2023.

[4] H. Zhou, C. Ibrahim, W. X. Zheng, and W. Pan, "Sparse Bayesian deep learning for dynamic system identification," *Automatica*, vol. 144, p. 110489, Oct. 2022.

[5] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste, "Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks.," *J. Mach. Learn. Res.*, vol. 22, no. 241, pp. 1–124, 2021.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[8] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," 2017.

[9] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2017.

[10] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019.

[11] M. Winter and C. Breitsamter, "Nonlinear identification via connected neural networks for unsteady aerodynamic analysis," *Aerospace Science and Technology*, vol. 77, pp. 802–818, 2018.

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[13] H. Robinson, E. Lundby, A. Rasheed, and J. T. Gravdahl, "A novel corrective-source term approach to modeling unknown physics in aluminum extraction process," *arXiv*, 2022.

[14] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, p. 109597, 2021.