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

Approximate Model Predictive Control of Switched Affine Systems using Multitask Learning with Safety and Stability Guarantees

Faiq Ghawash[1], Morten Hovd[1] and Brad Schofield[2]

Abstract— **We study the problem of designing an approximate model predictive control (MPC) for discrete time switched affine systems. The MPC design for the switched affine system requires an online solution of a mixed integer program. However, the combinatorial nature of the mixed integer problems might require a large computational time limiting its applicability in real time scenarios. To this end, we propose a framework based on the multitask learning paradigm to approximate the solution of mixed integer MPC for switched affine systems. We also provide a computational method to overapproximate the reachable sets of the closed-loop system that helps to analyze the safety and stability of the system under the influence of the learned controller. Once trained offline, the resulting controller results in a solver free approach especially suited for implementation on resource constrained embedded hardware. We demonstrate the efficacy of the approach on a real world example of an induced draft cooling tower.**

I. Introduction

Discrete time switched affine systems (SASs) are an important class of hybrid systems involving multiple operational modes where each mode is governed by a distinct set of affine difference equations. The switching between different operational modes takes place through internally generated or externally applied control signals. Switched affine systems are used to model complex engineering systems in the process control industry [1], power electronics applications [2], automotive systems [3], etc. The control design for such systems poses a challenging task owing to the simultaneous determination of the operational mode and actuator signals.

Among different control strategies, MPC has been widely adopted to ensure the safety and efficient operation of the switched systems [4] [5]. The finite horizon optimal control problem (OCP) associated with the MPC is usually cast as a mixed integer program (MIP) to simultaneously determine the best operational mode and optimal actuator signals. The OCP is required to be solved at each sampling instant, however, the combinatorial nature of the MIP might result in a large memory footprint and computation time limiting its applicability in real time scenarios. An alternative strategy involves explicit hybrid MPC which relies on the offline computation of the control law in which the MPC controller is equivalently expressed as a lookup table of linear gains [6]. However, as the number of binary variables and constraints increases, the computational and memory requirements for explicit MPC become prohibitively expensive [7]. With the recent success of learning based algorithms in different application areas, multitask learning (MTL) based on deep neural network (DNN) can provide an effective framework for approximating the solution of MPC for the SASs.

In recent years, DNN has been adopted as an effective strategy to approximate the solution of different variations of MPC. In [8] the authors utilize DNN to approximate the solution to the unconstrained nonlinear MPC. The case of constrained linear MPC has been studied in [9] where the constraint satisfaction is achieved by projecting the output of the DNN onto a set that ensures the state of the system remains within an appropriately designed invariant set. Similarly, reachability analysis has also been employed for analyzing the constraint satisfaction and stability of the systems controlled by DNN [10] [11] [12]. DNN has also been used to approximate the solution of robust nonlinear MPC [13]. In the case of hybrid systems involving continuous and binary variables, different machine learning algorithms (random forest, decision tree, support vector machines) are employed to learn the predictor for binary variables resulting in a partial approximation of the solution to hybrid MPC [14] [15] [16]. In addition, the strategy of employing continuous relaxation for binary variables and utilizing distinct neural networks for predicting continuous and binary variables has also found application in approximating solutions for hybrid MPC [17][18]. Despite the recent advancement in machine/deep learning assisted solutions for hybrid MPC, a systematic and scalable approach for approximating solutions to MPC for switched/hybrid systems is required to handle the challenge of simultaneous determination of binary and continuous input variables. Furthermore, safety and stability guarantees are pivotal for the deployment of the learned controllers in the production environment.

In this work our contributions are as follows: i) we propose an MTL framework based on DNN to approximate the solution of MPC for switched dynamical systems. The proposed MTL framework can learn to simultaneously determine the best operational mode and optimal actuator signals and is well suited for implementation on resource constrained embedded hardware. ii) A mixed integer linear encoding of the closed-loop system is presented which is then used for the computation of polyhedral over approximation of the reachable sets of the closed-loop system. iii) Based on the reachability analysis, sufficient conditions are derived for the safety and stability of switched affine system under the proposed multitask DNN controller.

The rest of the paper is organized as follows: Section 2

[1]The authors are with the Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway {faiq.ghawash,morten.hovd}@ntnu.no

[2]The author is with Beams Department, Industrial Controls Systems Group at CERN, Meyrin, Switzerland brad.schofield@cern.ch

provides a brief overview of the MPC design for discrete time SASs. Section 3 presents the proposed multitask DNN architecture and provides details on approximating the solution of MPC. Section 4 discusses the mixed integer linear encoding of the proposed multitask DNN controller. Section 5 formulates the mixed integer linear program (MILP) for computing the over approximation of the reachable sets for the closed-loop system and provides sufficient conditions for the safety and stability of the system. Section 6 provides an illustrative example and section 7 concludes the paper.

II. Problem Formulation

A. Discrete Time Switched Affine System

We consider a discrete time non-autonomous switched affine system comprising of L subsystems with the following state space realization:

$$x_{k+1} = A_{\sigma_k} x_k + B_{\sigma_k} u_k + g_{\sigma_k} \tag{1}$$

where k is the sampling instant, $x \in \mathbb{R}^{n_x}$ is the state of the system, $u \in \mathbb{R}^{n_u}$ is the actuator input signals and $\sigma \in \Sigma = \{1, \cdots, L\}$ is a piecewise constant function representing the active operational mode. For a given operational mode ($\sigma = l$), the pair (A_l, B_l, g_l) is the l^{th} subsystem where A_l is the state transition matrix, B_l is the input matrix and g_l is the constant vector representing the offset term of the l^{th} subsystem. The input u is subjected to box constraints whereas the state of the system x belongs to the compact polyhedral set,

$$u \in \mathcal{U} = \{u \in \mathbb{R}^{n_u} | u^{min} \leq u \leq u^{max}\} \tag{2}$$

$$x \in \mathcal{X} = \{x \in \mathbb{R}^{n_x} | C_x x \leq E_x\}. \tag{3}$$

In this work, we consider SASs which can have multiple equilibrium points, and the desired equilibrium point doesn't necessarily coincide with the equilibrium point of any isolated subsystems [19] [20]. This class of SASs can model systems of practical significance [1] [2], and the control of such systems typically requires stabilization of the system state to some set around the desired equilibrium point.

B. Model Predictive Control of Switched Affine System

In recent years, MPC has been widely adopted for designing the control of non-autonomous switched affine systems [4], [5]. MPC can provide several advantages in terms of handling actuator and state constraints and can ensure optimal performance with respect to the defined performance measure. In an MPC, the operational mode and the actuator signals are simultaneously computed by solving a finite time optimal control problem (OCP) given as follows:

$$\begin{aligned}
\min_{\mathbf{u},\boldsymbol{\sigma}} \quad & \|Px_N\|_p + \sum_{k=0}^{N-1} \|Qx_k\|_p + \|Ru_k\|_p \\
\text{s.t.} \quad & x_{k+1} = A_{\sigma_k} x_k + B_{\sigma_k} u_k + g_{\sigma_k},\ \forall k \in \mathcal{K}_{[0,N-1]} \\
& u_k \in \mathcal{U},\ \forall k \in \mathcal{K}_{[0,N-1]} \\
& \sigma_k \in \Sigma,\ \forall k \in \mathcal{K}_{[0,N-1]} \\
& x_k \in \mathcal{X},\ \forall k \in \mathcal{K}_{[0,N-1]} \\
& x_N \in \mathcal{X}_N \\
& x_0 = x^{init}.
\end{aligned} \tag{4}$$

Here N represents the prediction horizon. For $p = 2$, $\|Oo_k\|_p = o_k^T O o_k$ where $O = \{P, Q, R\}$ and P, Q ≥ 0 and $R > 0$. Similarly, for $p = 1, \infty$, we have $\|Oo_k\|_p = \|Oo_k\|_{1,\infty}$ where P, Q and R are nonsinglular. The solution to the finite time optimal control problem is a duplet consisting of i) finite sequence of active operational modes ($\boldsymbol{\sigma} = \{\sigma_0^*, \cdots, \sigma_{N-1}^*\}$) and ii) sequence of the actuator input signals ($\mathbf{u} = \{u_0^*, \cdots, u_{N-1}^*\}$). The switching times can be indirectly obtained from the finite sequence of active operational mode. The solution to the above OCP can be obtained by casting the problem as a MIP capable of simultaneously determining the best operational mode and optimal actuator signals [5] [1].

III. MPC Solution Approximation using Multitask Learning

A. Multitask Learning

Multitask learning (MTL) is a machine learning technique where a predictor is trained to perform multiple tasks simultaneously. In the MTL framework, the predictor can leverage knowledge across different tasks to improve the performance. The simultaneous learning of multiple tasks can be beneficial in situations where there is limited data available for a single task, or when multiple tasks are related and can benefit from shared representations. Multitask learning has been widely adopted in many application areas including computer vision [21], natural language processing [22], biomedical imaging, etc. MTL provides significant advantages in terms of computational complexity, memory footprint and is well suited for implementation on embedded hardware [23]. Several algorithms are available to solve the multitask learning problem, however, we restrict our focus to MTL framework based on DNN. Interested readers are referred to [24] for a review of available algorithms and their different application areas.

The MPC design for the switched systems requires the simultaneous determination of the best operational mode and optimal actuator signals to achieve the desired control objective. Such a problem inherently admits a multitask formulation where one task is to decide the best operational mode and the other tasks require to determine the optimal actuator signals. These tasks are often strongly coupled as the actuator behavior is usually strongly dependent on the operating mode of the switched system. Hence, the multitask learning framework can be effectively applied to approximate the MPC controller for the switched systems.

B. Data Generation

Multitask DNN usually rely on the labeled data sets (i.e features vector and target values) to learn their tasks. In control applications, the *features vector* usually contains information about the state of the system [1] whereas the *target values* contains the corresponding information of optimal control signals (i.e best operational mode and optimal

[1]Feature vector can also contain information about the measured/predicted disturbances and other related information about the system.

actuator signals in case of switched systems). The feature vectors can be generated using different sampling techniques including Sobol sequences [25], random sampling, etc to effectively cover the domain of operation of the system with a limited number of samples. On the other hand, the target values can either be generated using expert knowledge or by solving MIP problems offline against each generated feature vector. The data set for training a DNN can be obtained by collecting the following pairs $\Gamma = \{(x_1, \{u_1^0, \sigma_1^0\}), \cdots, (x_n, \{u_n^0, \sigma_n^0\}) \subseteq \mathbb{R}^{n_x} \times \mathbb{R}^{n_u + n_L}\}$. It is important to note that the operational mode of the switched system (σ^0) is represented using one-hot encoding in the dataset (i.e a vector containing the probability of each operational mode).

C. Proposed Multitask Deep Neural Network Architecture

In this subsection, we cover the preliminaries on DNNs and propose our multitask DNN architecture to approximate the solution of MPC for switched dynamical systems. The building block of a DNN is a neuron which is a function that computes the weighted sum of its inputs and adds a bias term to it ($\bar{y}(x) = \sum_{i=1}^{n_x} w_i x_i + c$). The resultant value is passed through an activation function (ψ) to get the output of the neuron.

$$y(x) = \psi(\sum_{i=1}^{n_x} w_i x_i + c) = \psi(w^T x + c) \tag{5}$$

where, $x \in \mathbb{R}^{n_x}$, $w \in \mathbb{R}^{n_w}$ and $c \in \mathbb{R}$ represents the input vector, weight vector and bias respectively. Different activation functions including *Linear, ReLU, Tanh, Sigmoid and Softmax* can be employed. The choice of an activation function for a neuron in a DNN is mainly based on the problem being addressed, the desired output range, and the ease of calculation of the function's derivative for back-propagation purposes. Different activation functions used in this work are summarized in Table I.

TABLE I: Different activation functions

Activation Function	**Formula**
Linear	$\psi_{lin}(\bar{y}) = \bar{y}$, $\bar{y} \in \mathbb{R}$
ReLU	$\psi_{relu}(\bar{y}) = \max(0, \bar{y})$, $\bar{y} \in \mathbb{R}$
Softmax	$\psi_{softmax}(h_j) = \frac{exp(h_j)}{\sum_{l=1}^{L} exp(h_l)}$, $h \in \mathbb{R}^{n_L}$

DNN consists of a network of neurons and has been successfully employed to approximate complex nonlinear functions [26]. The proposed DNN architecture for approximating the solution of MPC for switched systems is shown in Fig 1. It consists of an *input layer*, shared and task-specific *hidden layers*, and task specific *output layers* labeled as $\mathcal{L} = \{\mathcal{L}_0, \{\mathcal{L}_1, \cdots, \mathcal{L}_{n_s}\}, \{\mathcal{L}_1^\sigma, \cdots, \mathcal{L}_{n^\sigma}^\sigma\}, \{\mathcal{L}_1^u, \cdots, \mathcal{L}_{n^u}^u\}\}$. Here n_s is the number of shared layers, n^σ and n^u represents the number of task specific layers for mode prediction and actuator signals determination tasks respectively. Different variables associated with the r^{th} neuron in a given layer $\mathcal{L}_p^q$ are denoted by $(.)_r^{\mathcal{L}_p^q}$. The number of shared and task-specific hidden layers and the number of neurons in each of these layers can be varied based on the complexity of the problem. For $\mathcal{Y}^{\mathcal{L}_0} = x$, $\mathcal{Y}^{\mathcal{L}_0^\sigma} = \mathcal{Y}^{\mathcal{L}_{n_s}}$, $\mathcal{Y}^{\mathcal{L}_0^u} = \mathcal{Y}^{\mathcal{L}_{n_s}}$,

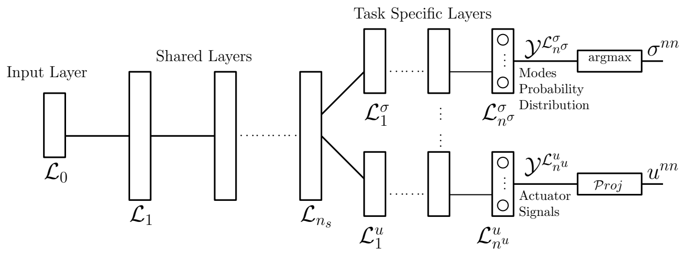

Fig. 1: Multitask DNN architecture for approximating the solution of MPC for switched systems.

the multitask DNN architecture can be defined using the following equations:

$$\begin{aligned}
\mathcal{Y}^{\mathcal{L}_i} &= \psi_{relu}(0, W^{\mathcal{L}_i}\mathcal{Y}^{\mathcal{L}_{i-1}} + b^{\mathcal{L}_i}), \forall i \in \{1, \cdots, n_s\} \\
\mathcal{Y}^{\mathcal{L}_i^\sigma} &= \psi_{relu}(0, W^{\mathcal{L}_i^\sigma}\mathcal{Y}^{\mathcal{L}_{i-1}^\sigma} + b^{\mathcal{L}_i^\sigma}), \forall i \in \{1, \cdots, n^\sigma - 1\} \\
\mathcal{Y}^{\mathcal{L}_i^u} &= \psi_{relu}(0, W^{\mathcal{L}_i^u}\mathcal{Y}^{\mathcal{L}_{i-1}^u} + b^{\mathcal{L}_i^u}), \forall i \in \{1, \cdots, n^u - 1\} \\
\mathcal{Y}^{\mathcal{L}_{n^\sigma}^\sigma} &= \psi_{softmax}(W^{\mathcal{L}_{n^\sigma}^\sigma}\mathcal{Y}^{\mathcal{L}_{n^\sigma-1}^\sigma}) \\
\mathcal{Y}^{\mathcal{L}_{n^u}^u} &= \psi_{lin}(W^{\mathcal{L}_{n^u}^u}\mathcal{Y}^{\mathcal{L}_{n^u-1}^u})
\end{aligned} \tag{6}$$

The active operational mode and the actuator input signals can be obtained from the DNN output as follows: $\sigma = \arg\max(\mathcal{Y}^{\mathcal{L}_{n^\sigma}^\sigma})$, $u = \mathcal{Y}^{\mathcal{L}_{n^u}^u}$. Here, $\mathcal{Y}^{\mathcal{L}_i}$, $\mathcal{Y}^{\mathcal{L}_i^\sigma}$ and $\mathcal{Y}^{\mathcal{L}_i^u}$ represents the values of the neurons at the shared and task specific layers. The nonlinear activation function ψ_{relu} must be applied elementwise whereas $\psi_{softmax}$ converts the output of the mode prediction task layer into modes probability distribution. $W^{\mathcal{L}_i}, W^{\mathcal{L}_i^\sigma}, W^{\mathcal{L}_i^u}$ and $b^{\mathcal{L}_i}, b^{\mathcal{L}_i^\sigma}, b^{\mathcal{L}_i^u}$ are the weight matrices and bias vectors of appropriate dimensions for the shared and task specific layers. Also note that, $\theta = (W^{\mathcal{L}_i}, W^{\mathcal{L}_i^\sigma}, W^{\mathcal{L}_i^u}, b^{\mathcal{L}_i}, b^{\mathcal{L}_i^\sigma}, b^{\mathcal{L}_i^u})$ are learnable parameters that must be optimized against the dataset. Finally, the neural network control law can be written as follows:

$$[\sigma^{nn}, u^{nn}] = \pi_{nn}(x|\theta) \tag{7}$$

where π_{nn} represents the proposed multitask DNN controller. Moreover, a projection operator is considered in the loop to ensure input constraint satisfaction. The projected input is defined as follows:

$$\hat{u}^{nn} = \mathcal{P}roj(u^{nn}) = \begin{cases} u^{max}, & u^{nn} \geq u^{max} \\ u^{min}, & u^{nn} \leq u^{min} \\ u, & otherwise \end{cases} \tag{8}$$

The projected neural network is denoted as $[\sigma^{nn}, \hat{u}^{nn}] = \hat{\pi}_{nn}(x|\theta)$. Finally, we note that the resulting controller is a nonsmooth nonlinear function mainly due to the presence of nonlinear activations and an input projection operator.

D. Loss Function and Training of Neural Network

In order to jointly learn the tasks of mode selection and actuator signal determination, an appropriate loss function must be defined. We adopt cross-entropy loss for predicting the mode probability distribution and mean squared error loss is used for actuator signals determination. The loss function is defined as the weighted sum of the mean squared error and cross-entropy loss functions.

$$\mathcal{L}(\theta) = \mathcal{L}_{CE} + \mathcal{L}_{MSE}$$
$$= -\alpha \sum_{i=1}^{n}\sum_{j=1}^{L} \sigma_{i,j}^{0} log(y_{i,j}^{\mathcal{L}_{n^\sigma}^{\sigma}}) + \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n_u} \beta_j (u_{i,j}^{nn} - u_{i,j}^{0})^2 \quad (9)$$

Here, α is the weight associated with the cross-entropy loss, whereas β_j is the weight associated with the mean squared error loss function for each actuator and n is the number of training examples. CE loss is better suited for predicting the modes probability distribution owing to its better convergence properties and its ability to penalize more strongly for incorrect mode predictions [27]. The unknown parameters θ are estimated from the data set Γ by minimizing the loss function:

$$\min_{\theta} \mathcal{L}(\theta) \quad (10)$$

Different training algorithms can be used to learn the parameters of the proposed multitask DNN including gradient descent, stochastic gradient descent, etc. Note that, the training process for estimating the unknown parameters in multitask DNN is computationally expensive and must be done offline. *Once trained, the multitask DNN output simply corresponds to the evaluation of the learned function w.r.t to the input feature vector*. It is easy to see that, the multitask DNN architecture can be easily extended to approximate the controller for other classes of hybrid systems involving binary and continuous variables with appropriately designed loss functions.

IV. Mixed Integer Encoding of the Closed-Loop System

In this section, our main aim is to present the mixed integer linear encoding of the learned controller and the closed-loop system which will play a key role in analyzing the safety and stability of the system.

A. Encoding of the Multitask DNN

The encoding of the learned controller is mainly based on modeling the input-output behavior of each neuron in the multitask DNN architecture. The main challenge lies in dealing with the nonlinearities arising in the form of different activation functions and projection operator. Our approach is based on transforming such nonlinearities and discontinuities into mixed integer linear inequalities using the McCormick relaxation (also known as Big-M technique) [28]. We start by encoding the input-output behavior of neurons with the *ReLU* activation present in the hidden layers of multitask DNN architecture.

Proposition 1: $y_r^{\mathcal{L}_p^q} = \max\left(0, \bar{y}_r^{\mathcal{L}_p^q}\right)$ *iff the constraints (11a) to (11c) hold:*

$$0 \le y_r^{\mathcal{L}_p^q} \le M_{r,1}^{\mathcal{L}_p^q}(1 - \lambda_{r,1}^{\mathcal{L}_p^q}) \quad (11a)$$
$$\bar{y}_r^{\mathcal{L}_p^q} \le y_r^{\mathcal{L}_p^q} \le \bar{y}_r^{\mathcal{L}_p^q} + M_{r,2}^{\mathcal{L}_p^q}(1 - \lambda_{r,2}^{\mathcal{L}_p^q}) \quad (11b)$$
$$\lambda_{r,1}^{\mathcal{L}_p^q} + \lambda_{r,2}^{\mathcal{L}_p^q} = 1 \quad (11c)$$

The proof can be found in Appendix. $\lambda_{r,1}^{\mathcal{L}_p^q}$ and $\lambda_{r,2}^{\mathcal{L}_p^q}$ are the binary variables associated with the r^{th} neuron of the $\mathcal{L}_p^q$ layer in DNN architecture. At the output layer of mode selection task, the neurons have *Softmax* activation function that involves the computation of the exponential function. Exponential functions can't be exactly encoded as mixed integer linear inequalities. However, the following result helps to omit the construction of the *Softmax* activation function in the analysis under specific conditions.

Proposition 2 ([29])**:** *Given a multitask DNN with the output layer of the modes probability distribution prediction task having Softmax activation function and a constant $\gamma > 0$, then $\forall r, s \in \Sigma$*

$$y_r^{\mathcal{L}_{n^\sigma}^{\sigma}} \ge \gamma y_s^{\mathcal{L}_{n^\sigma}^{\sigma}} \iff \bar{y}_r^{\mathcal{L}_{n^\sigma}^{\sigma}} \ge ln(\gamma) + \bar{y}_s^{\mathcal{L}_{n^\sigma}^{\sigma}} \quad (12)$$

Proposition 2 implies that when the analysis doesn't require the exact values and depends mainly on the ratio ordering of the values of neurons in the output layer of the mode selection task. Then, the property to be analyzed can be written in terms of the output values of neurons without the *Softmax* activation function. Consequently, the active operational mode can be obtained by finding the maximum value index among the output values of neurons without the *Softmax* activation function. Then the argmax function can be modeled using mixed integer linear inequalities.

Proposition 3: $\sigma^{nn} = \arg\max(\bar{y}_1^{\mathcal{L}_{n^\sigma}^{\sigma}}, \cdots, \bar{y}_L^{\mathcal{L}_{n^\sigma}^{\sigma}})$ *iff the constraints (13a) to (13b) hold:*

$$\bar{y}_r^{\mathcal{L}_{n^\sigma}^{\sigma}} \le y^{aux} \le \bar{y}_r^{\mathcal{L}_{n^\sigma}^{\sigma}} + M_r^{\mathcal{L}_{n^\sigma}^{\sigma}}(1 - \lambda_r^{\mathcal{L}_{n^\sigma}^{\sigma}}),\ \forall r \in \Sigma \quad (13a)$$
$$\sum_{r=1}^{L} \lambda_r^{\mathcal{L}_{n^\sigma}^{\sigma}} = 1 \quad (13b)$$

The proof can be found in the Appendix. Here, y^{aux} represents the auxiliary continuous variable and the activated binary variable represents the operational mode of the system. Moreover, the output layer of the actuator signals determination task utilizes a linear activation function and hence can be simply modeled using the linear equality constraint.

$$u_r^{nn} = y_r^{\mathcal{L}_{n^u}^{u}}, \forall r \in \{1, \cdots, n_u\} \quad (14)$$

Finally, the input projection operators of the form (7) can either be rewritten as additional layers of the DNN [11] [12] or can be directly encoded as mixed integer linear inequalities.

Proposition 4: $\hat{u}^{nn} = \mathcal{P}roj(u^{nn})$ *iff the constraints (15a) to (15i) hold:*

$$u_j^{nn} \le u_j^{min} + M_{j,1}^{proj}(1 - \lambda_{j,1}^{proj}) \quad (15a)$$
$$u_j^{nn} \ge u_j^{max} - M_{j,2}^{proj}(1 - \lambda_{j,2}^{proj}) \quad (15b)$$
$$u_j^{nn} \ge u_j^{min} - M_{j,3}^{proj}(1 - \lambda_{j,3}^{proj}) \quad (15c)$$
$$u_j^{nn} \le u_j^{max} + M_{j,3}^{proj}(1 - \lambda_{j,3}^{proj}) \quad (15d)$$
$$\hat{m}_{j,1}^{proj}(1 - \lambda_{j,1}^{proj}) \le \hat{u}_j^{nn} - u_j^{min} \le \hat{M}_{j,1}^{proj}(1 - \lambda_{j,1}^{proj}) \quad (15e)$$
$$\hat{m}_{j,2}^{proj}(1 - \lambda_{j,2}^{proj}) \le \hat{u}_j^{nn} - u_j^{max} \le \hat{M}_{j,2}^{proj}(1 - \lambda_{j,2}^{proj}) \quad (15f)$$
$$\hat{m}_{j,3}^{proj}(1 - \lambda_{j,3}^{proj}) \le \hat{u}_j^{nn} - u_j^{nn} \le \hat{M}_{j,3}^{proj}(1 - \lambda_{j,3}^{proj}) \quad (15g)$$
$$\lambda_{j,1}^{proj} + \lambda_{j,2}^{proj} + \lambda_{j,3}^{proj} = 1 \quad (15h)$$
$$\forall j \in \{1, \cdots, n_u\} \quad (15i)$$

This completes the encoding of the projected DNN that plays a key role in approximating the reachable sets for the closed-loop system which in turn helps to analyze the various properties of the closed-loop system.

B. Encoding of the Closed-Loop System

In order to analyze the system level properties under the projected neural network controller, the closed-loop behavior must be modeled. Given the system (1) controlled by the DNN controller ($\hat{\pi}^{nn}$), the closed-loop system can be written as follows:

$$x_{k+1} = f^{\hat{\pi}^{nn}}(x_k) = A_{\sigma_k^{nn}} x_k + B_{\sigma_k^{nn}} \hat{u}_k^{nn} + g_{\sigma_k^{nn}} \tag{16}$$

The closed-loop system can then be modeled using the following mixed integer linear inequalities:

Proposition 5: *$f^{\hat{\pi}^{nn}}(x_k) = A_{\sigma_k^{nn}} x_k + B_{\sigma_k^{nn}} \hat{u}_k^{nn} + g_{\sigma_k^{nn}}$ iff the constraints (17a) and (17b) holds:*

$$x_{k+1} - (A_j x_k + B_j \hat{u}_k^{nn} + g_j) \le M_j(1 - \lambda_j^{\mathcal{L}^{\sigma}_{n_\sigma}}), \forall j \in \Sigma \tag{17a}$$
$$x_{k+1} - (A_j x_k + B_j \hat{u}_k^{nn} + g_j) \ge m_j(1 - \lambda_j^{\mathcal{L}^{\sigma}_{n_\sigma}}), \forall j \in \Sigma \tag{17b}$$

The proof can be found in the Appendix. Note that, the binary variables used to model the argmax function are used for activating a certain subsystem in (1). Finally, we emphasize that appropriate values of the Big-M constants (m and M) plays an important role in avoiding infeasibility and weak relaxations of the mixed integer program. The values of these constants can be obtained using interval arithmetics [29].

V. Reachability Analysis of the Closed-loop System

Reachability analysis has been widely adopted in various fields to verify the safety and correctness of designed systems. In the context of dynamic systems, reachability analysis based on MIP can be employed to analyze the safety and stability properties of the closed-loop system [10] [11]. Reachability of the closed-loop system involves the computation of all possible states of the system that could be reached from a given set of initial conditions ($\mathcal{X}_{init}$). Given a closed-loop system (16), the forward reachable set at time K from a given set $\mathcal{X}_{init}$ is defined by the following function composition:

$$\mathcal{R}_K(\mathcal{X}_{init}) := f_K^{\hat{\pi}^{nn}} \circ \cdots \circ f^{\hat{\pi}^{nn}}(\mathcal{X}_{init}) \tag{18}$$

The problem of computing the exact reachable sets for the switched affine system is computationally expensive. However, the over approximation of reachable sets can be computed using polyhedral approximations.

A. Over Approximation of the Reachable Sets

Next, we propose a technique based on solving MILP for computing the over approximation of reachable sets for the closed-loop system. First, we define the notion of support functions, which plays a key role in constructing polyhedral approximation of the reachable sets. The support function of a compact nonempty set Ω is a function $\rho_\Omega(d) : \mathbb{R}^{n_d} \to \mathbb{R}$ that attributes to a direction d the scalar value s.t

$$\rho_\Omega(d) = \max_{z \in \Omega} d^T z. \tag{19}$$

$\rho_\Omega(d)$ actually determines the position of the halfspace i.e

$$\mathcal{H}_d = \{z \in \mathbb{R}^{n_z} | d^T z \le \rho_\Omega(d)\} \tag{20}$$

that touches and contains Ω. Given a set $\mathcal{D} = \{d_1, \cdots, d_{n_d}\}$ containing a finite number of direction vectors, an over-approximation of set Ω is the polyhedral obtained from the intersection of halfspaces i.e

$$\lceil \Omega \rceil = \bigwedge_{i \in D} \{z \in \mathbb{R}^{n_z} | d_i^T z \le \rho_\Omega(d_i)\} \tag{21}$$

where $\lceil \Omega \rceil$ is the over approximation of the set Ω (i.e $\Omega \subseteq \lceil \Omega \rceil$). $\lceil \Omega \rceil$ is also referred to as the template polyhedron with template direction specified by $\mathcal{D}$. The polyhedral approximation using support functions provides a unique advantage over traditional techniques where the approximation can be refined on demand by adding additional directions. *One can interpret evaluating support functions as the lazy, on-demand, construction of a template polyhedron* [30]. Given the finite set of template directions $\mathcal{D}$, the support functions of the K-step reachable set ($\rho_{\mathcal{R}_K}(d_i)$) of the closed-loop system from a given set $\mathcal{X}_{init}$ can be determined by solving the following MILP for each direction d_i:

$$\max_{x_0, \boldsymbol{\lambda}, \mathbf{x}, \mathbf{y}, \mathbf{u}} \quad d_i^T x_K \tag{22a}$$
$$\text{s.t.} \quad [\sigma_k^{nn}, u_k^{nn}] = \pi^{nn}(x_k), \; \forall k \in \mathcal{K}_{[0,\cdots,K-1]} \tag{22b}$$
$$\hat{u}_k = \mathcal{P}roj(u_k^{nn}), \; \forall k \in \mathcal{K}_{[0,\cdots,K-1]} \tag{22c}$$
$$x_{k+1} = A_{\sigma_k^{nn}} x_k + B_{\sigma_k^{nn}} \hat{u}_k^{nn} + g_{\sigma_k^{nn}} \tag{22d}$$
$$\forall k \in \mathcal{K}_{[0,\cdots,K-1]} \tag{22e}$$
$$x_0 \in \mathcal{X}_{init} \tag{22f}$$

Note that (22b), (22c), (22d) must be encoded using the mixed integer linear inequalities as presented in the previous section. The computation of the support functions using (22) enables the construction of a polyhedral over approximation of the K-step reachable set of the closed-loop system as follows:

$$\lceil \mathcal{R}_K(\mathcal{X}_{init}) \rceil = \bigwedge_{i \in \mathcal{D}} \{x_k \in \mathbb{R}^n | d_i^T x_k \le \rho_{\mathcal{R}_k}(d_i)\} \tag{23}$$

The computation of $\lceil \mathcal{R}_K \rceil$ allows analyzing the state constraint satisfaction and stability of the closed-loop system. We note here that, the complexity of MIP increases exponentially with the increase in the number of binary variables. Hence, for $K > 1$ the $\lceil \mathcal{R}_K(\mathcal{X}_{init}) \rceil$ can also be computed in a recursive fashion.

$$\lceil \mathcal{R}_K(\mathcal{X}_{init}) \rceil = \lceil \mathcal{R}_1(\lceil \mathcal{R}_i \rceil) \rceil, \lceil \mathcal{R}_0 \rceil = \mathcal{X}_{init}, \forall i \in \{0, \cdots, K-1\} \tag{24}$$

B. Safety (Closed-Loop State Constraint Satisfaction)

The approximation of the MPC using DNN doesn't guarantee state constraint satisfaction in the design phase. However, state constraint satisfaction guarantees can be obtained by leveraging the reachability analysis of the closed-loop system.

Theorem 1: *Given the switched system (1) with initial conditions $x_0 \in \mathcal{X}$ controlled by a projected DNN controller ($\hat{\pi}^{nn}$) and an over-approximation of a 1-step reachable set ($\lceil \mathcal{R}_1(\mathcal{X}) \rceil$) obtained by solving (22) globally. If $\lceil \mathcal{R}_1(\mathcal{X}) \rceil \subseteq \mathcal{X}_{in}$, then the state constraints are satisfied for all time $k \geq 0$.*

Proof: Let x_k represent the state of the closed-loop system at any arbitrary time step k, where $x_k \in \mathcal{X}$. By definition of 1-step reachable set of the closed-loop system, the next state of the closed-loop system lies in the over approximation of the 1-step reachable set (i.e $x_{k+1} = f^{\hat{\pi}_{nn}}(x_k) \in \lceil \mathcal{R}_1(\mathcal{X}) \rceil$). Since the over approximation of the reachable set is contained in the state constraint set (i.e $\lceil \mathcal{R}_1(\mathcal{X}) \rceil \subseteq \mathcal{X}$), therefore we conclude that the next state also lies within the state constraint set (i.e $x_{k+1} \in \mathcal{X}$). By observing the initial state of the closed-loop system lies in the state constraint set (i.e $x_0 \in \mathcal{X}$), it follows that the state of the closed-loop system will remain within the state constraint set for all time steps $k \geq 0$. Hence, state constraints are satisfied for all time $k \geq 0$ $\Box$.

C. Practical Asymptotic Stability of Switched Affine Systems

Next, our main aim is to analyze the stability of the switched affine system under the projected neural network controller. When the subsystems have distinct equilibria and the desired equilibrium point doesn't coincide with the equilibrium point of any isolated subsystems, bringing the system trajectories within a certain bound from the desired equilibrium point and keeping them within that bound thereafter is of practical significance. Such a concept has been formally termed as ϵ-practical asymptotic stability [19]. Without loss of generality, let x_d be the desired equilibrium point:

Definition 1 (ϵ-practical asymptotic stability)**:** *Given $\epsilon > 0$ and $\delta < \epsilon$, the system (1) is said to be ϵ-practically asymptotic stable around x_d under the projected neural network ($\hat{\pi}^{nn}$) if:*

- *(ϵ-practical stability) there exists a δ such that $\|x_k - x_d\| < \epsilon$ whenever $x_0 \in \mathcal{X}$ satisfies $\|x_0 - x_d\| < \delta$.*
- *(ϵ-attractivity) for every $x_0 \in \mathcal{X}$ there exists a $T = T(x_0) > 0$ such that $\|x_k - x_d\| < \epsilon$ for any $k \geq T$.*

Here, it is important to emphasize that ϵ-attractivity doesn't necessarily guarantee ϵ-stability. There may exist a trajectory that starts at x_0 with $\|x_0 - x_d\| < \delta$, violates the condition $\|x_k - x_d\| < \epsilon$ for some time, but eventually converges to $\|x_k - x_d\| < \epsilon$. Such a trajectory would still fulfill the requirements of ϵ-attractivity, but not ϵ-practical stability. Hence ϵ-practical asymptotic stability requires the fulfillment of both ϵ-practical stability and ϵ-attractivity properties.

Theorem 2: *Given the switched system (1) with initial conditions $x_0 \in \mathcal{X}$ controlled by a projected DNN controller ($\hat{\pi}^{nn}$), a bounded ball centered at the desired operating point $\mathcal{B}[x_d, \epsilon]$ and an over-approximation of reachable sets ($\lceil \mathcal{R}_K(\mathcal{X}) \rceil$, $\lceil \mathcal{R}_1(\mathcal{B}[x_d, \epsilon]) \rceil$) computed by solving (22) globally. Then the system (1) under the controller ($\hat{\pi}^{nn}$) is ϵ-practical asymptotic stable:*

- $\lceil \mathcal{R}_K(\mathcal{X}) \rceil \subseteq \mathcal{B}[x_d, \epsilon]$
- $\lceil \mathcal{R}_1(\mathcal{B}[x_d, \epsilon]) \rceil \subseteq \mathcal{B}[x_d, \epsilon]$.

Proof: In order to prove the theorem, first we need to show that the trajectories starting from any admissible initial condition $x_0 \in \mathcal{X}$ will converge to $\mathcal{B}[x_d, \epsilon]$ within a finite number of time steps. Secondly, we need to demonstrate all the trajectories starting within $\mathcal{B}[x_d, \epsilon]$ will remain within the $\mathcal{B}[x_d, \epsilon]$ for all future time steps. The proof of the second part is similar to the proof in Theorem-1, and hence omitted. For the first part, let x_K be the state of the system at the K^{th} time step starting from any admissible initial condition $x_0 \in \mathcal{X}$. By definition of K-step reachable set of the closed-loop system, the state of the closed-loop system at the K^{th} time step lies within the over approximation of the K-step reachable set (i.e $x_K \in \lceil \mathcal{R}_K(\mathcal{X}) \rceil$). Since the over approximation of the K-step reachable set is contained in $\mathcal{B}[x_d, \epsilon]$, therefore we conclude that the state of the system will always lie within $\mathcal{B}[x_d, \epsilon]$ after K time steps. Hence, the system trajectories starting from any admissible initial condition $x_0 \in \mathcal{X}$ enters the $\mathcal{B}[x_d, \epsilon]$ at or before K^{th} time step and remains there for all future time steps $\Box$.

VI. Simulation Results

In this section, we show the efficacy of the proposed multitask DNN to approximate the solution of the MPC for an induced draft cooling tower (IDCT). IDCT can be operated in three different operational modes namely active cooling, free cooling, and bypass to regulate the outlet water temperature to the desired setpoint. Active cooling mode is governed by non-autonomous affine dynamics whereas free cooling and bypass modes are governed by autonomous affine dynamics. In the active cooling mode, the fan speed (with minimum speed constraint) can be modulated to achieve the required heat rejection requirements whereas the fan remains off in free cooling (natural draft) and bypass modes. Each operational mode has a different cooling capacity and depending on the supply water temperature and the weather conditions, the desired cooling capacity might not coincide with the cooling capacity of any operational mode. More details can be found in [1] [31].

The multitask DNN controller for the IDCT comprises of three hidden layers, each comprising eight neurons. Among these layers, one is shared, while the remaining two are dedicated to mode selection and fan speed control respectively. Figure (2), (3), (4) shows the performance of the multitask DNN for regulating the outlet water temperature of the IDCT to $22.5°C$ from different initial conditions. The supply and wet bulb temperatures are $27°C$ and $16.5°C$ respectively. The state constraint set is given by $\mathcal{X} = \{x \in \mathbb{R} | 20 \leq x \leq 25\}$ and the bounded ball is centered at the setpoint $\mathcal{B}[22.5, 0.2]$. The 1-step reachable sets from $\mathcal{X}$ and $\mathcal{B}$ are computed to be $\mathcal{R}_1(\mathcal{X}) = \{x \in \mathbb{R} | 20.58 \leq x \leq 24.68\}$, $\mathcal{R}_1(\mathcal{B}) = \{x \in \mathbb{R} | 22.38 \leq x \leq 22.67\}$ respctively. Similarly, the 25-step reachable set is computed to be $\mathcal{R}_{25}(\mathcal{B}) = \{x \in \mathbb{R} | 22.40 \leq x \leq 22.52\}$. Note that

$\mathcal{R}_1(\mathcal{X}) \subset \mathcal{X}$, $\mathcal{R}_1(\mathcal{B}) \subset \mathcal{B}$ and $\mathcal{R}_{25}(\mathcal{X}) \subset \mathcal{B}$ which satisfies the conditions for the state constraint satisfaction as well as the ϵ-practical stability of the closed-loop system.

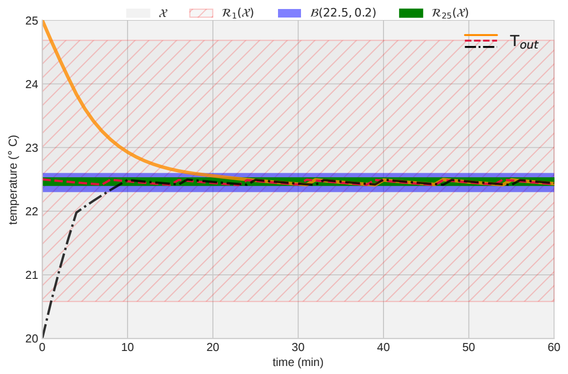

Fig. 2: Regulation of outlet water temperature to $22.5°C$ under the multitask DNN controller from different initial conditions.

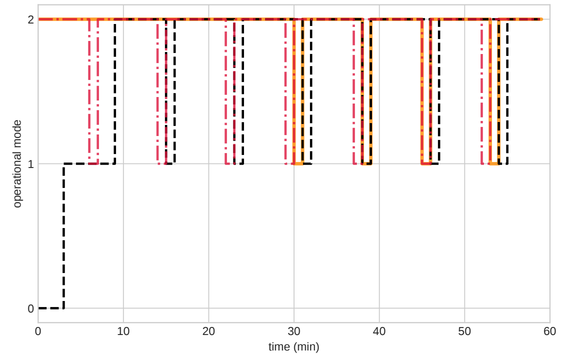

Fig. 3: Mode switching under multitask DNN controller. $0, 1$ and 2 correspond to bypass, free cooling and active cooling modes respectively.

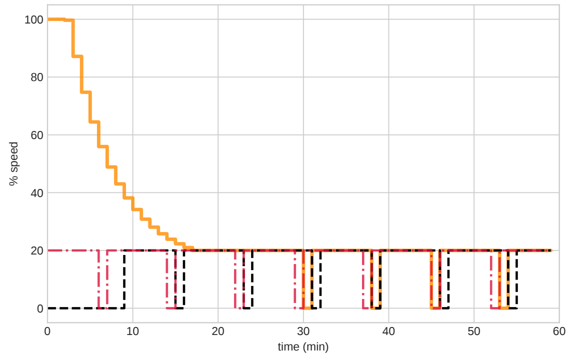

Fig. 4: Percentage fan speed.

VII. CONCLUSIONS

In this paper, we proposed a multitask DNN architecture capable of simultaneously determining the operational mode and actuator signals for controlling a non-autonomous switched affine system. A computation method based on MILP was presented to compute the over approximation of the reachable sets of the closed-loop system which were then used to derive the sufficient conditions for state constraint satisfaction and practical stability of the closed system. In the future, we plan to extend the formulation to include the dwell time constraints and investigate the approaches for systematic refinement of the over approximation of the reachable sets.

REFERENCES

[1] F. Ghawash, M. Hovd, and B. Schofield, "Optimal control of induced draft cooling tower using mixed integer programming," in *2021 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2021, pp. 214–219.

[2] L. N. Egidio, H. R. Daiha, G. S. Deaecto, and J. C. Geromel, "Dc motor speed control via buck-boost converter through a state dependent limited frequency switching rule," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 2072–2077.

[3] S. Di Cairano, J. Doering, I. V. Kolmanovsky, and D. Hrovat, "Model predictive control of engine speed during vehicle deceleration," *IEEE transactions on control systems technology*, vol. 22, no. 6, pp. 2205–2217, 2014.

[4] L. J. Bridgeman, C. Danielson, and S. Di Cairano, "Stability and feasibility of mpc for switched linear systems with dwell-time constraints," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 2681–2686.

[5] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[6] A. Bemporad, J. Baillieul, and T. Samad, "Explicit model predictive control." 2015.

[7] G. Cimini and A. Bemporad, "Exact complexity certification of active-set methods for quadratic programming," *IEEE Transactions on Automatic Control*, vol. 62, no. 12, pp. 6094–6109, 2017.

[8] T. Parisini and R. Zoppoli, "A receding-horizon regulator for nonlinear systems and a neural approximation," *Automatica*, vol. 31, no. 10, pp. 1443–1451, 1995.

[9] S. Chen, K. Saulnier, N. Atanasov, D. D. Lee, V. Kumar, G. J. Pappas, and M. Morari, "Approximating explicit model predictive control using constrained neural networks," in *2018 Annual American control conference (ACC)*. IEEE, 2018, pp. 1520–1527.

[10] S. Dutta, S. Jha, S. Sankaranarayanan, and A. Tiwari, "Learning and verification of feedback control systems using feedforward neural networks," *IFAC-PapersOnLine*, vol. 51, no. 16, pp. 151–156, 2018.

[11] B. Karg and S. Lucia, "Stability and feasibility of neural network-based controllers via output range analysis," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 4947–4954.

[12] H. Hu, M. Fazlyab, M. Morari, and G. J. Pappas, "Reach-sdp: Reachability analysis of closed-loop systems with neural network controllers via semidefinite programming," in *2020 59th IEEE conference on decision and control (CDC)*. IEEE, 2020, pp. 5929–5934.

[13] S. Lucia and B. Karg, "A deep learning-based approach to robust nonlinear model predictive control," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 511–516, 2018.

[14] D. Masti and A. Bemporad, "Learning binary warm starts for multiparametric mixed-integer quadratic programming," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 1494–1499.

[15] D. Masti, T. Pippia, A. Bemporad, and B. De Schutter, "Learning approximate semi-explicit hybrid mpc with an application to microgrids," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 5207–5212, 2020.

[16] Y. Löhr, M. Klaučo, M. Fikar, and M. Mönnigmann, "Machine learning assisted solutions of mixed integer mpc on embedded platforms," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 5195–5200, 2020.

[17] B. Karg and S. Lucia, "Deep learning-based embedded mixed-integer model predictive control," in *2018 european control conference (ecc)*. IEEE, 2018, pp. 2075–2080.

[18] L. Markolf and O. Stursberg, "Learning-based optimal control of constrained switched linear systems using neural networks." in *ICINCO*, 2021, pp. 90–98.

[19] X. Xu and P. J. Antsaklis, "Practical stabilization of integrator switched systems," in *Proceedings of the 2003 American Control Conference, 2003.*, vol. 4. IEEE, 2003, pp. 2767–2772.

[20] G. S. Deaecto and L. N. Egidio, "Practical stability of discrete-time switched affine systems," in *2016 European Control Conference (ECC)*. IEEE, 2016, pp. 2048–2053.

[21] R. Ranjan, S. Sankaranarayanan, C. D. Castillo, and R. Chellappa, "An all-in-one convolutional neural network for face analysis," in *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*. IEEE, 2017, pp. 17–24.

[22] R. Collobert and J. Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning,” in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 160–167.

[23] K.-H. Thung and C.-Y. Wee, “A brief review on multi-task learning,” *Multimedia Tools and Applications*, vol. 77, no. 22, pp. 29 705–29 725, 2018.

[24] P. Vafaeikia, K. Namdar, and F. Khalvati, “A brief review of deep multi-task learning and auxiliary task learning,” *arXiv preprint arXiv:2007.01126*, 2020.

[25] H. Niederreiter, “Low-discrepancy and low-dispersion sequences,” *Journal of number theory*, vol. 30, no. 1, pp. 51–70, 1988.

[26] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[27] P. Golik, P. Doetsch, and H. Ney, “Cross-entropy vs. squared error training: a theoretical and experimental comparison.” in *Interspeech*, vol. 13, 2013, pp. 1756–1760.

[28] I. E. Grossmann, “Review of nonlinear mixed-integer and disjunctive programming techniques,” *Optimization and engineering*, vol. 3, pp. 227–252, 2002.

[29] C.-H. Cheng, G. Nührenberg, and H. Ruess, “Maximum resilience of artificial neural networks,” in *Automated Technology for Verification and Analysis: 15th International Symposium, ATVA 2017, Pune, India, October 3–6, 2017, Proceedings 15*. Springer, 2017, pp. 251–268.

[30] M. Althoff and G. Frehse, “Combining zonotopes and support functions for efficient reachability analysis of linear systems,” in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 7439–7446.

[31] F. Ghawash, M. Hovd, and B. Schofield, “Capacity control of induced draft cooling tower using two stage optimization,” in *2021 25th International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE, 2021, pp. 485–490.

Appendix

A. Proof of Proposition 1

Proof: **Case i)** Assume that $\bar{y}_r^{\mathcal{L}_p^q} < 0$, then the feasibility of the constraints in (11) requires setting $\lambda_{r,1}^{\mathcal{L}_p^q} = 1$ and $\lambda_{r,2}^{\mathcal{L}_p^q} = 0$. When $\lambda_{r,1}^{\mathcal{L}_p^q} = 1$, the constraints in $(11a)$ are binding and together imply that $y_r^{\mathcal{L}_p^q} = 0$. As $\lambda_{r,2}^{\mathcal{L}_p^q} = 0$, the lower bound in $11(b)$ is less restrictive than the lower bound in $11(a)$ as $\bar{y}_r^{\mathcal{L}_p^q} < 0$. Similarly, for a suitably large value of the constant $M_{r,1}^{\mathcal{L}_p^q}$, the upper bound in $11(b)$ is less restrictive than the upper bound in $11(a)$, hence the constraints in $11(b)$ are not binding. We thus have $\bar{y}_r^{\mathcal{L}_p^q} < 0 \implies \lambda_{r,1}^{\mathcal{L}_p^q} = 1$, $\lambda_{r,2}^{\mathcal{L}_p^q} = 0 \implies y_r^{\mathcal{L}_p^q} = 0$. Similarly, from the definition we have $\bar{y}_r^{\mathcal{L}_p^q} < 0 \implies y_r^{\mathcal{L}_p^q} = 0$, which requires setting $\lambda_{r,1}^{\mathcal{L}_p^q} = 1$, $\lambda_{r,2}^{\mathcal{L}_p^q} = 0$ and hence the $11(a)$ to $11(c)$ holds. **Case ii)** Assume $\bar{y}_r^{\mathcal{L}_p^q} > 0$, then the feasibility of the constraints in (11) requires setting $\lambda_{r,1}^{\mathcal{L}_p^q} = 0$, $\lambda_{r,2}^{\mathcal{L}_p^q} = 1$. When $\lambda_{r,2}^{\mathcal{L}_p^q} = 1$, the constraints in $(11b)$ are binding and together imply that $y_r^{\mathcal{L}_p^q} = \bar{y}_r^{\mathcal{L}_p^q}$. As $\lambda_{r,1}^{\mathcal{L}_p^q} = 0$, the lower bound in $11(a)$ is less restrictive than the lower bound in $11(b)$ as $\bar{y}_r^{\mathcal{L}_p^q} > 0$. Similarly, for a suitably large value of $M_{r,2}^{\mathcal{L}_p^q}$, the upper bound in $11(a)$ is less restrictive than the upper bound in $11(b)$, hence the constraints in $11(a)$ are not binding. We thus have $\bar{y}_r^{\mathcal{L}_p^q} > 0 \implies \lambda_{r,1}^{\mathcal{L}_p^q} = 0$, $\lambda_{r,2}^{\mathcal{L}_p^q} = 1 \implies y_r^{\mathcal{L}_p^q} = \bar{y}_r^{\mathcal{L}_p^q}$. The other side of the implication follows similar argument as in the Case i. **Case iii)** when $\bar{y}_r^{\mathcal{L}_p^q} = 0$, the constraints are satisfied with $y_r^{\mathcal{L}_p^q} = 0$ regardless of which binary variable is chosen to be activated.

B. Proof of Proposition 3

Proof: Without loss of generality, let's assume that $\bar{y}_i^{\mathcal{L}_{n^\sigma}^\sigma}$ be the maximum among the values of the neurons in the output layer (without *Softmax* activation) of modes selection task. The constraints feasibility in (13) require setting $\lambda_i^{\mathcal{L}_{n^\sigma}^\sigma} = 1$, $\lambda_j^{\mathcal{L}_{n^\sigma}^\sigma} = 0, \forall j \in \Sigma, j \neq i$. When $\lambda_i^{\mathcal{L}_{n^\sigma}^\sigma} = 1$, the constraints in $13(a_i)$ are binding and together imply that $y_{aux} = \bar{y}_i^{\mathcal{L}_{n^\sigma}^\sigma}$. As $\lambda_j^{\mathcal{L}_{n^\sigma}^\sigma} = 0, \forall j \in \Sigma, j \neq i$, the lower bounds in the constraints $13(a_j)$ are less restrictive than the lower bound in $13(a_i)$ as $\bar{y}_i^{\mathcal{L}_{n^\sigma}^\sigma} > \bar{y}_j^{\mathcal{L}_{n^\sigma}^\sigma}$. Similarly, for a suitably large values of the constants $M_j^{\mathcal{L}_{n^\sigma}^\sigma}$, the upper bounds in the constraints $13(a_j) \forall j \in \Sigma, j \neq i$ are less restrictive than the upper bound in $13(a_i)$, hence the constraints in $13(a_j)$ are not binding. Thus we have, $\bar{y}_i^{\mathcal{L}_{n^\sigma}^\sigma} > \bar{y}_j^{\mathcal{L}_{n^\sigma}^\sigma}, \forall j \in \Sigma, j \neq i \implies \lambda_i^{\mathcal{L}_{n^\sigma}^\sigma} = 1 \implies \sigma^{nn} = i$. Similarly, it is trivial to see that $\sigma^{nn} = i$ require setting $\lambda_i^{\mathcal{L}_{n^\sigma}^\sigma} = 1$ and $\lambda_j^{\mathcal{L}_{n^\sigma}^\sigma} = 0$ $\forall j \in \Sigma, j \neq i$ and hence the constraints in $(13a)$ and $(13b)$ holds.

C. Proof of Proposition 4

Proof: Case i) Assume $u_j^{nn} < u_j^{min}$, then the feasibility of the constraints in (15) requires setting $\lambda_{j,1}^{proj} = 1$, $\lambda_{j,2}^{proj} = 0$ and $\lambda_{j,3}^{proj} = 0$. When $\lambda_{j,1}^{proj} = 1$, the constraints in $(15e)$ are binding and together imply that $\hat{u}_j^{nn} = u_j^{min}$. As $\lambda_{j,2}^{proj} = 0$ and $\lambda_{j,3}^{proj} = 0$, the lower bounds in the constraints $15(f), 15(g)$ are less restrictive than the lower bound in $15(e)$ for suitably small values of $\hat{m}_{j,2}^{proj}$ and $\hat{m}_{j,3}^{proj}$. Similarly, the upper bound in $15(e)$ and $15(f)$ are less restrictive than the upper bound in $15(d)$ for suitably large values of $\hat{M}_{j,2}^{proj}$ and $\hat{M}_{j,3}^{proj}$. Hence the constraints in $15(f)$ and $15(g)$ are not binding. We thus have $u_j^{nn} < u_j^{min} \implies \lambda_{j,1}^{proj} = 1$, $\lambda_{j,2}^{proj} = 0$, $\lambda_{j,3}^{proj} = 0 \implies \hat{u}_j^{nn} = u_j^{min}$. Along the same lines one can show that $u_j^{nn} > u_j^{max} \implies \lambda_{j,1}^{proj} = 0$, $\lambda_{j,2}^{proj} = 1$, $\lambda_{j,3}^{proj} = 0 \implies \hat{u}_j^{nn} = u_j^{max}$ and $u_j^{min} \leq u_j \leq u_j^{max} \implies \lambda_{j,1}^{proj} = 0$, $\lambda_{j,2}^{proj} = 0$, $\lambda_{j,3}^{proj} = 1 \implies u_j^{nn} = u_j^{nn}$. Moreover, the other side of the implication follows a similar argument as in the proposition 1 and 3.

D. Proof of Proposition 5

Proof: When $\lambda_i^{\mathcal{L}_{n^\sigma}^\sigma} = 1$ and $\lambda_j^{\mathcal{L}_{n^\sigma}^\sigma} = 0 \; \forall j \in \Sigma, j \neq i$, the constraint in the $17(a_i)$ and $17(b_i)$ are binding and together imply that $x_{k+1} = (A_i x_k + B_i \hat{u}_k^{nn} + g_i)$. As $\lambda_j^{\mathcal{L}_{n^\sigma}^\sigma} = 0, \forall j \in \Sigma, j \neq i$, the lower bounds in the constraints $17(a_j)$ are less restrictive than the lower bound in $17(a_i)$ for suitable smaller values of the constants m_j. Similarly, for suitably large values of the constants M_j, the upper bounds in the constraints $17(a_j)$ are less restrictive than the upper bound in $17(a_i)$, hence the constraints in $17(a_j)$ are not binding. Thus we have, $\lambda_i^{\mathcal{L}_{n^\sigma}^\sigma} = 1$ and $\lambda_j^{\mathcal{L}_{n^\sigma}^\sigma} = 0$ $\forall j \in \Sigma, j \neq i \implies x_{k+1} = (A_i x_k + B_i \hat{u}_k^{nn} + g_i)$. Hence the state dynamics evolve using the i^{th} subsystem difference equations. The other side of the implication follows a similar argument as in the proposition 1 and 3.