9th ANKARA INTERNATIONAL AEROSPACE CONFERENCE **AIAC-2017-012**
20-22 September 2017 - METU, Ankara TURKEY

NEURAL NETWORKS BASED DYNAMICAL MODEL FOR VTOL AIRCRAFT

Metehan Yayla[*] and Tugba Tuncel[†]
Middle East Technical University
Ankara, Turkey

ABSTRACT

For the rotorcraft systems, system identification is a challenging problem since the aerodynamics, structure, and environment of the rotorcraft platform are highly coupled with each other. In this study, we propose a deep learning architecture with multiple semi-decoupled hidden layers. Network architecture is constructed for an unmanned VTOL aircraft platform. It's layer placements are constructed regarding the physical aspects of the motion of considered aerial vehicle. In the end, accurate deep learning model is obtained, and the findings are illustrated with simulations.

INTRODUCTION

Deep learning has gained significant importance with the advancing computer technology. With its increasing power, the applications of the deep learning algorithms become larger including the finger-print recognition [Wang, 2014], image classification for large dataset [Krizhevsky, 2012], language translation [Collobert, 2008], and system identification of dynamic systems [Ogunmolu, 2016]. In this study, we make use of the deep learning for the system identification of the novel vertical takeoff landing (VTOL) capable aircraft. The control allocation in the designed VTOL aircraft is quite distinct than the previous designs in the literature, it is harder to obtain a high fidelity model of the aircraft dynamics.

Especially for the rotorcraft systems, system identification becomes a challenging problem since the aerodynamics, structure, and environment of the rotorcraft platform are highly coupled with each other. In order to address this question, many works have been done related to system identification of rotorcraft systems [Sguanci, 2012; Wu, 2014; Mettler, 2000; Kaymak, 2013]. In older works, either time domain or frequency domain approaches with different optimization algorithms are used. But recently with the rise of Deep Neural Networks, researchers started using different type of Neural Networks to overcome rotorcraft system identification challenges.

Most of the research in this area are devoted to model of Helicopter Dynamics [Manso, 2015; Punjani, 2015; Abbeel, 2010]. In [Manso, 2015], the authors proposed a Support Vector Regression (SVR) algorithm to to predict pitch angle response of a helicopter to a longitudinal control input using advanced flight simulator data. Although model seems promising, it is just a simplification of complete model since it just considers longitudinal input and pitch angle. Entire dynamics of an aerobatic helicopter is considered in [Punjani, 2015]. A ReLU Network Model is developed. This network is a

[*]Grad. Res. Asst. in Department of Aerospace Engineering, Email: myayla@metu.edu.tr
[†]Grad. Res. Asst. in Department of Aerospace Engineering, Email: ttuncel@metu.edu.tr

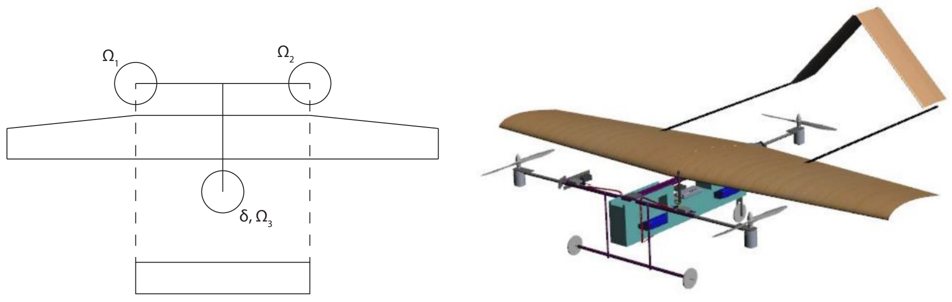

Figure 1: Conceptual and CAD Drawings of VTOL UAV[Onen, 2015]

combination of a Quadratic Lag Model which is a physics based model and a two layer neural network (NN). With their special initialization and optimization techniques, developed model match quite well with test data in all flight regimes. Since they are using quadratic lag model, parameters of this model should be selected carefully regarding physics of flight. To overcome this part, we developed a neural network, which does not require any physical model.

In this study, we propose a deep learning architecture with multiple semi-decoupled hidden layers to obtain an accurate dynamic model of the novel VTOL unmanned aircraft platform. It is known that some certain states and control inputs are dominant on the pitch channel, and some others are dominant on the roll and/or yaw channels. Further, it is also known that the yaw and roll motion are not so distinguishable, and they can be considered together in the lateral dynamics. Although, the lateral and longitudinal motions have weak couplings when the aircraft is modeled with simple dynamics, there may be strong couplings for the high fidelity models, as well. In our semi-decoupled multilayer architecture, all these information are embedded to the layer structure, and the number of neurons in each layer. Further, a new activation function is employed in the network. In the end, accurate deep learning model is obtained, and the findings are illustrated with simulations.

PROBLEM DESCRIPTION

The dynamics of the VTOL aircraft can be considered as:

$$x(k+1) = f(k, \tau(k), x(k), \gamma) \tag{1}$$

where $x(k) \in \Re^n$ is the state vector, $f(k, \tau(k), x(k), \gamma)$ is a mapping, $\tau(k) \in \Re^m$ with $m \leq n$ is the control input, $\gamma \in \Re^s$ is the time-independent parameters. For simplicity, the arguments 'k' denoting the time dependencies are dropped from the variables if they are not crucial for the formulation. The states $x(k)$, and the control inputs $\tau(k)$ are given as:

$$\begin{gathered} x = \begin{bmatrix} u & v & w & p & q & r & \theta & \phi \end{bmatrix}^T \\ \tau = \begin{bmatrix} \Omega_1 & \Omega_2 & \Omega_3 & \delta \end{bmatrix}^T \end{gathered} \tag{2}$$

where u, v, w are body velocities in X, Y, Z directions (resp.), p, q, r are angular rate components about X, Y, Z directions (resp.), and θ, ϕ are the pitch and roll attitudes (resp.). Ω_i is the rpm control for the i^{th} motor, δ is the tilt control for the aft motor. Conceptual and CAD drawings of the VTOL unmanned aerial vehicle (UAV) are given in figure 1.

The function $f(\cdot, \cdot, \cdot, \cdot)$ which is a mapping for the ideal behavior of the rotorcraft is highly nonlinear, and its structure is highly complex. Thus, it is impossible to model the rotorcraft exactly with the known modeling and identification techniques. In this study, we will approximate the mapping $f : \Re \times \Re^m \times \Re^n \times \Re^s \to \Re^n$ with a deep learning model $\hat{f} : \Re \times \Re^m \Re^n \to \Re^n$ such that the overall dynamics becomes:

$$x(k+1) = \hat{f}(k, \tau(k), x(k); W) + \varepsilon(k, x(k)) \tag{3}$$

where $x(k) \in \mathcal{D}_x \subset \Re^n$, $\|\varepsilon(k, x(k))\| \leq \varepsilon_0 \; \forall k \geq 0$, $\forall x(k) \in \mathcal{D}_x$, and W is the NN model parameters. The set $\mathcal{D}_x$ can be enlarged by using as many distinct samples as possible in the training. Therefore, without loss of generality, we assume that the set $\mathcal{D}_x$ is sufficiently large. The upper bound ε_0 for the residual can be made arbitrarily small by increasing the dimension of the deep learning model. The inputs for the deep learning model will be the time instant k for determining the width of the window, the system states $x(k)$, and the control inputs $\tau(k)$. Then, the outputs are the system states at the next time instant $x(k+1)$.

DEEP LEARNING MODEL

ANN Architecture

Three minor motions of the aircraft about the principal axes X, Y, Z are called roll, pitch, and yaw. Therefore, the first hidden layer is divided into three channels representing these motions. In many aerial vehicles, the general motion of the aircraft can be collected under two major titles which are longitudinal and lateral-directional motions. The longitudinal motion is highly dominated by the pitch channel. On the other hand, the lateral-directional motion consists of strong couplings of yaw and roll channels. Regarding these information, pitch channel in the NN architecture is directly connected to the longitudinal channel whereas roll and yaw channel are combined in the lateral channel. Eventually, longitudinal and lateral channels construct the general motion of the aircraft which is progressed the second hidden layer. The NN architecture is given in figure 2. We will call our NN architecture as ***metaNet*** for the rest of the paper.

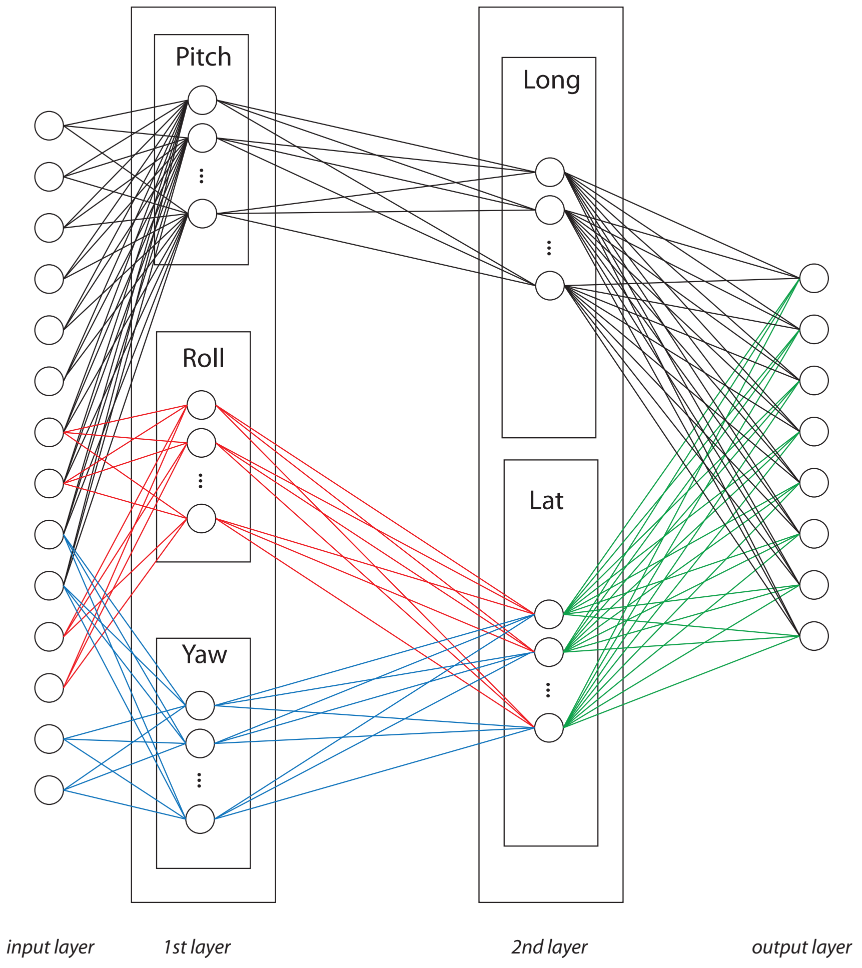

Figure 2: Neural Network Architecture

Input Structure

The inputs for the ***metaNet*** consisting of system states and control inputs are divided into three channels which are corresponding to the pitch, yaw, and roll. The dominant NN model inputs for pitch channel are given as:

$$x_p = \begin{bmatrix}\Omega_1 & \Omega_2 & \Omega_3 & \delta & q & \theta & u & w\end{bmatrix} \tag{4}$$

where x_p is the set of NN model inputs passing through the pitch channel in the first hidden layer. Similarly, NN model inputs for the roll and yaw channels are determined as:

$$\begin{aligned} x_r &= \begin{bmatrix}\Omega_1 & \Omega_2 & p & \phi\end{bmatrix} \\ x_y &= \begin{bmatrix}\Omega_1 & \Omega_2 & \Omega_3 & \delta & r & v\end{bmatrix}\end{aligned} \tag{5}$$

where x_r and x_y denote the set of inputs for the roll and yaw channels, respectively. Since we are interested in the solution of the difference equation (3), we pay attention to the system state history and control input history to obtain the system states at the latter time instant. Let ζ be the set of dummy variables $\zeta = \{\Omega_1, \Omega_2, \Omega_3, \delta, u, v, w, p, q, r, \theta, \phi\}$, and ζ_i be the i^{th} element of the set ζ. Let us take the dummy variable $\zeta_5 = u$, for example, to give concrete example. As we said, time histories of the control inputs and the system states are taken into account. Then, we state our stacked elements in a vector $\xi_i = [u(k-d), u(k-d+1), \ldots, u(k)]$ for the i^{th} variable $\zeta_i = u$. We have the same stacking story for every variable ζ_i for $i = 1, 2, \ldots, 12$. In each NN input sets, there exist significant variations in the order of elements. For instance, the control input for the motors are given in $\Omega_i \in [1100, 1700]$ whereas the body velocity components u, v, w are such that $u, v, w \in [-30, 30]$. Thus, we employ mean-max normalization to each NN model inputs. That is:

$$\bar{\xi}_i = \frac{\xi_i - \mathrm{mean}(\xi_i)}{\max\left(|\xi_i|\right)} \tag{6}$$

where the operator $|\cdot|$ takes the absolute value of each element of ξ_i which is the normalization of ξ_i. As a result, intervals for each parameters are transformed to the $[-1, 1]$; i.e. $\xi_i \in [-1, 1]$. Eventually, the NN model inputs become:

$$\begin{aligned} X_p &= \begin{bmatrix}\xi_1 & \xi_2 & \xi_3 & \xi_4 & \xi_5 & \xi_7 & \xi_9 & \xi_{11}\end{bmatrix} \\ X_r &= \begin{bmatrix}\xi_1 & \xi_2 & \xi_8 & \xi_{12}\end{bmatrix} \\ X_y &= \begin{bmatrix}\xi_1 & \xi_2 & \xi_3 & \xi_4 & \xi_6 & \xi_{10}\end{bmatrix}\end{aligned} \tag{7}$$

where X_p, X_r, X_y are the stacked NN model inputs for the pitch, roll, and yaw channels, respectively. The label for the j^{th} sample is the vector system states of j^{th} at time instant $k+1$; that is $Y_j = \begin{bmatrix}\zeta_5(k+1) & \zeta_6(k+1) & \ldots & \zeta_{12}(k+1)\end{bmatrix}$ where ζ_i's are belonging to j^{th} sample.

Activation Function

As an activation function, we developed a modified version of Rectified Linear Unit (ReLU) which is called Biased Leaky Rectified Linear Unit (BL-ReLU). This function is constructed by considering the problem in our hand. As an biological motivation, the neuron generates an action potential or '*spike*' when sufficient input is received [Yi, 2015]. In Artificial Neural Networks, activation function plays the role of spike in real neurons. ReLU activation function, spikes when its input is greater than zero. But for our problem in hand using ReLU neglects the negative input which is important for us. Because of this reason, our activation function works in absolute region which means it activates neurons if functions absolute input is greater than some threshold λ which creates nonlinearity and using different slopes α, β for negative and positive regions are enabled to increase the nonlinearity. In figure 3, the activation function BL-ReLU is illustrated. For our problem since both negative and positive regions have the same priority, activating neurons in this regions using BL-ReLU function still preserves its biological meaning.

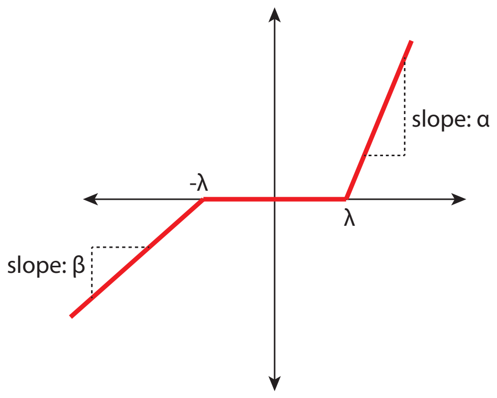

Figure 3: Biased Leaky Rectified Linear Unit (BL-ReLU)

Cost Function and Optimization (Training)

In this work, we are using mean square error with L_2 regularization with constant regularization parameter γ.

$$L(y,\widehat{y};W) = \frac{1}{N}\sum_{1}^{N}\left\|Y-\widehat{Y}\right\|^2 + \frac{\gamma}{N}\sum_{1}^{l}\left\|W^{(l)}\right\|^2 \tag{8}$$

where $W^{(l)}$ is the parameters at the layer l with $l = 1, 2, 3$, $\widehat{Y}$ is the estimation of the labels Y.

For optimization, cost function (8) is minimized using gradient descent algorithm. For parameter update, constant learning rate Γ is used.

$$\min_{W} L(y,\widehat{y};W) \tag{9}$$

Cross-Validation and Testing

First, the dataset is randomly mixed. Then, the entire dataset is divided into three portions. The first portion contains $\%60$ of the dataset, and is assigned as train set, X_{train}. Both cross-validation and test sets share the remaining $\%40$ portion, equally. Thus, both X_{cv} and X_{test} contain $\%20$ portion of the full data. Then, the hypothesis $\hat{f}(k,x;W)$ is trained using training set X_{train} as explained in the optimization part. The trained parameters θ are evaluated on the cross-validation set X_{cv}. Based on the cross-validation error, design parameters such as learning rate Γ, regularization parameter γ, order of initial uniformly distributed random weights, ***metaNet*** neuron dimensions, and BL-ReLU parameters (α, β, λ) are tuned. Thereafter, the final tuned parameters are evaluated on the test set X_{test} to obtain the generalization error.

Different from the literature, we apply BL-ReLU activation to the estimations with BL-ReLU parameters $\alpha = \beta = 1$, and $\lambda_0 = 0.01$. With this activation, we allow the ***metaNet*** to make errors up to $\%1$ percent. Implementation of such a *tunnel approach* improves the generalized success by decreasing the overfit. In figure 4, the notion behind the tunnel approach is illustrated. In physical

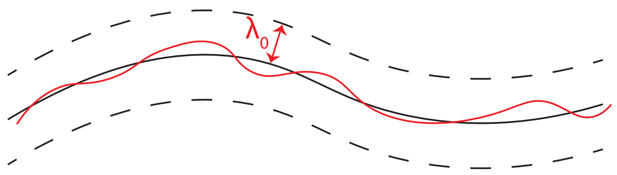

Figure 4: Tunnel Design

sense, we allow the estimations to move freely within the tunnel, and do not force them to be exactly equal to the measurements. This makes sense since the measurements themselves are not the ideal states due to sensor noise. Then, the $\%1$ error, can be related to the signal-to-noise ratio (SNR).

SIMULATION RESULTS

The ***metaNet*** is implemented using MATLAB. Experiments were run on dataset explained in input structure, and cross-validation and testing parts. Initially, the design parameters learning rate Γ, regularization parameter γ, number of neurons, positive and negative slopes of activation function α, β, order of initial weights, and the iteration number are set heuristically for the training purpose. Then using cross-validation set, all the parameters are tuned one by one.

Since Neural Networks are very sensitive to initial weights, first the initialization of weights is considered. The NN weights are initialized by random numbers between $[-1, 1]$. With this initialization, cost function diverged due to large gradients in itself. Because of this reason, we scaled the weight initializations with powers of number of training samples. As the scaling power increases cross-validation error kept decreasing but training error showed small oscillatory behavior which can be explained by randomness of initialization. Although smallest CV error is reached at power 0.7, 0.6 is taken as power since the network starts diverging beyond the scaling power of 0.7. In figure 5, the improvements achieved with initial weight scaling is illustrated.

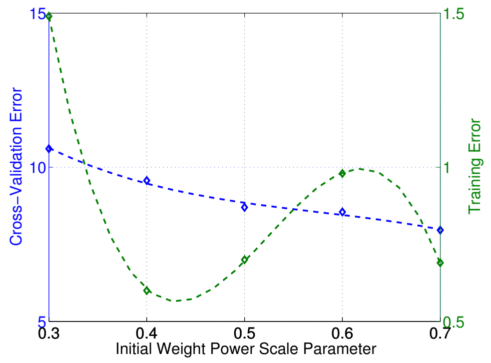

Figure 5: Power

Secondly, the effects of slopes in the positive and negative region in BL-ReLU activation function are investigated. In our physical problem, both positive and negative values of the system states and the control inputs have the same importance. This is because the negative sign just induces the motion in reverse direction. Therefore, we intuitively convinced ourselves that the slopes in positive and negative region should be the same. Because of this notion, we add deadzone to the activation function to add nonlinearity. Next, we verified our intuitions and discussions on the slopes of arms of the BL-ReLU activation function by tuning these parameters. Previously, we expressed our motivation on the arms of the BL-ReLU which should have the same priority. Hence, during the tuning procedure, we take the raio between these two slopes to inspect the effect of linearity and nonlinearity. Changing this raio, the change of training error and cross-validation error are observed, and given in figure 6.

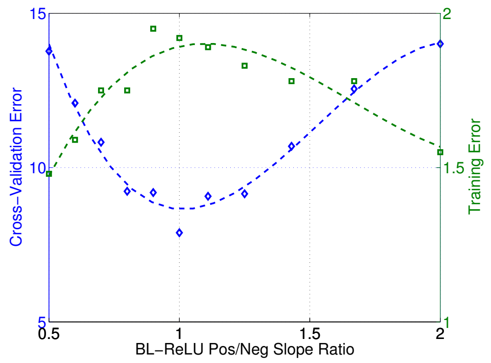

Figure 6: Slope Ratio

After several trials, the number of neurons are chosen to be $nn_{pitch} = 32$, $nn_{roll} = 16$, $nn_{yaw} = 16$, $nn_{long} = 32$, and $nn_{lat} = 32$. These neuron numbers are determined considering the dimensions of each channel. For instance, the pitch channel takes more inputs than the roll channel which makes the pitch channel more prone to larger couplings. In order to handle these kind of couplings, the number of neurons are related to the number of inputs for the corresponding layer.

From the figure 6, it is concluded that the cross-validation error decreases but training error increases as slope ratio approaches to 1. This behavior can be explained with overfitting. As we change slopes in negative and positive regions with respect to each other, the nonlinearity of the NN model increases, as well. That means the capacity of the NN model increases, and this makes our network more prone to overfitting. In the end, both of the slopes are chosen to be the same, and the deadzone modification in BL-ReLU is considered sufficient to have nonlinear decision boundaries.

Next, the learning rate is taken into account. Initial learning rate is set to 4.10^{-4}, then the behavior of the cost function during iterations are observed. Since it showed oscillations during iterations, learning rate decreased. This procedure is repeated until the optimum convergence rate of $\Gamma = 10^{-4}$ is reached. For the total iteration number, we did the similar thing, as well. When the progress in the cost function becomes smaller and smaller, we decided to break the iteration, and picked that value for our number of iterations.

Finally, the regularization parameter is taken into account. As we employ regularization to the cost function, we do not observe significant improvements on the cross-validation error. In fact, this implies that the NN model is not overfitting and works at the required level of capacity. Considering the neuron number we have used in the NN model, and the activation function BL-ReLU, it makes sense not to have both overfitting and underfitting. That is, the model is nonlinear having sufficient capacity, not more than enough. However, depending on the future input data, model may experience overfitting problem. So, we employ relatively small regularization to the cost function in order to stay on the safe side.

The comparison of the trained model and the NN model on the test set is illustared in figure 7.

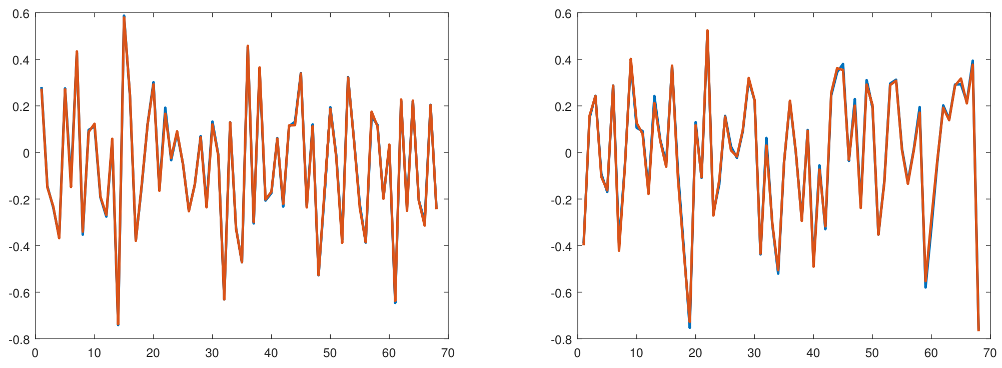

Figure 7: Comparison of Sample States (orange:NN Model, blue: Flight Test)

CONCLUSION

In this work, we introduce the Neural Network architecture with multiple semi-decoupled hidden layers and the novel BL-ReLU activation function. We further emonstrated its performance using the novel designed VTOL unmanned aircraft platform. Beside, we showed how using different parameters in the BL-ReLU activation function increases the capacity of model. NN model and the flight test data is compared, and the result is pretty much satisfactory. As a future work for complete model of VTOL aircraft, network should be trained with wider range flight regimes, and the optimization of the network parameters should be carried accordingly.

References

Abbeel, P. and Coates, A. and Ng, A. (2010) *Autonomous helicopter aerobatics through apprenticeship learning*, The International Journal of Robotics Research, 29(13):1608–1639, 2010

Collobert, R. and Weston, J. (2008) *A unified architecture for natural language processing: Deep neural networks with multitask learning*, 25th international conference on Machine learning, pp 160–167. ACM, 2008

Kaymak, S. (2013) *Evaluation of rotorcraft system identification approaches*, Master's thesis, Middle East Technical University, 2013

Krizhevsky, A. and Sutskever, I. and Hinton, G. (2013) *Imagenet classification with deep convolutional neural networks*, Advances in neural information processing systems, pp. 1097–1105, 2012

Manso, S. (2015) *Simulation and system identification of helicopter dynamics using support vector regression*, The Aeronautical Journal, 119(1222):1541–1560, 2015

Mettler, B. and Kanade, T. and Tischler, M. B. (2000) *System identification modeling of a model-scale helicopter*, Carnegie Mellon University, The Robotics Institute, 2000

Ogunmolu, O. and Gu, X. and Jiang, S. and Gans, N. *Nonlinear systems identification using deep dynamic neural networks*, arXiv preprint arXiv:1610.01439, 2016

Onen, A. S. (2015) *Modeling and controller design of a VTOL air vehicle*, Master's thesis, Middle East Technical University, 2015

Punjani, A. and Abbeel, P. (2015) *Deep learning helicopter dynamics models*, IEEE International Conference on Robotics and Automation (ICRA), pp 3223–3230, 2015

Sguanci, M. and Bergamasco, M. and Lovera, M. (2012) *Continuous-time model identification for rotorcraft dynamics*, System Identification, 16(1):816–821, 2012

Wang, R. and Han, C. and Wu, Y. and Guo, T. (2014) *Fingerprint classification based on depth neural network*, arXiv preprint:1409.5188, 2014

Wu, W. (2014) *Identification method for helicopter flight dynamics modeling with rotor degrees of freedom*, Chinese Journal of Aeronautics, 27(6):1363–1372, 2014

Yi, G.-S. and Wang, J. and Tsang, K.-M. and Wei, X.-L. and Deng, B. (2015) *Biophysical insights into how spike threshold depends on the rate of membrane potential depolarization in type i and type ii neurons*, PloS one, 10(6):e0130250, 2015