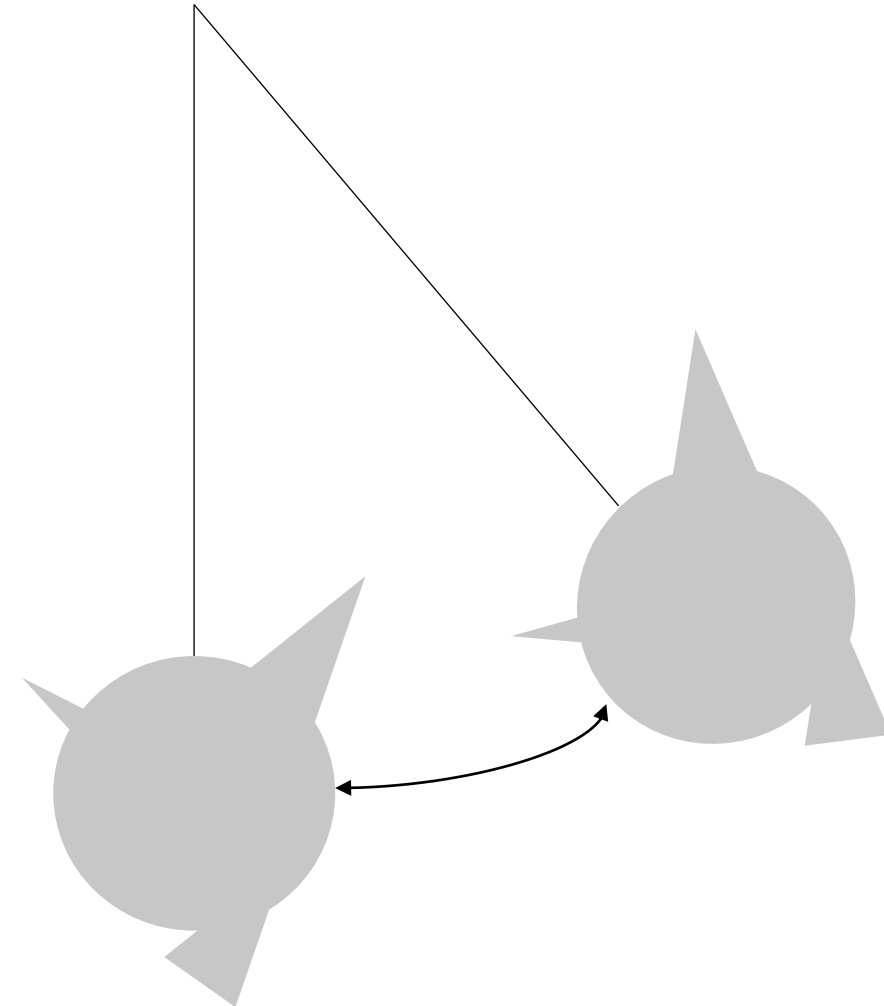


# Recurrent Neural Networks (RNN) for Modeling of Nonlinear Systems



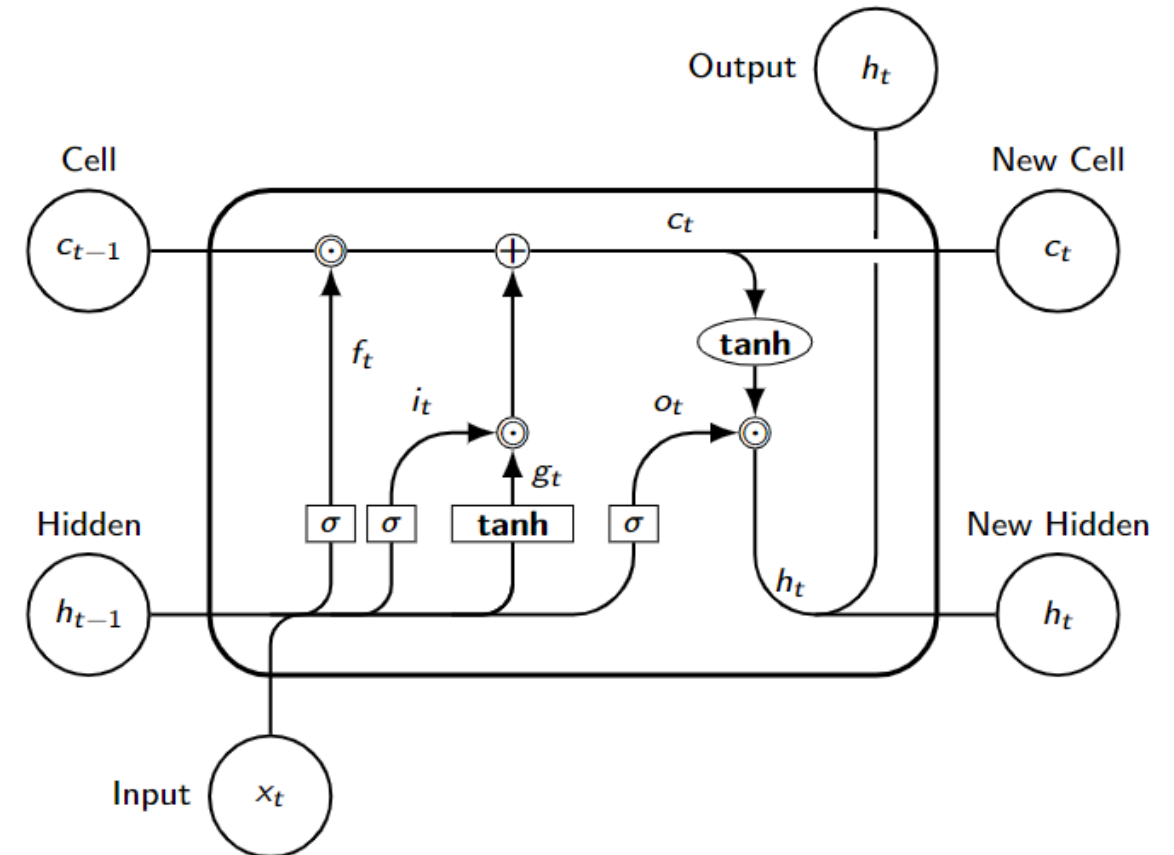
Sebastian Hirt | 19.01.2023  
Lehrstuhl für Regelungstechnik

- Problem:
  - Complex System without analytic form
  - Example: Pendulum with unknown friction model
- Solution
  - Train a model with measured data
  - Prediction of future states with model



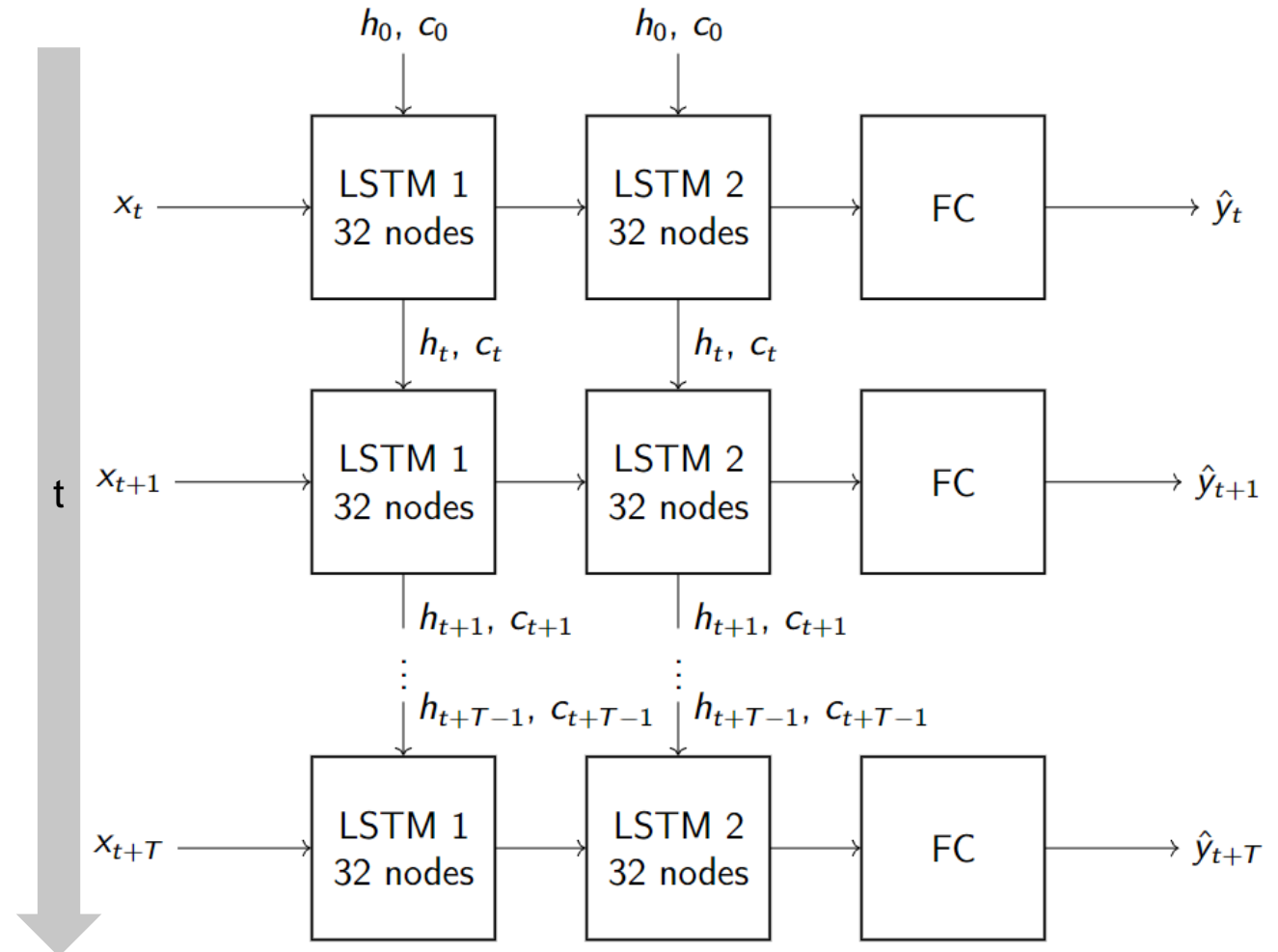
# Methods

- Long-term dependency through cell state
- Passing of new initial value to next timestep internally
- Gates with trainable weights
- Tanh activation function



Adapted from (Yu, Y., Si, X., Hu, C., & Zhang, J., 2019)

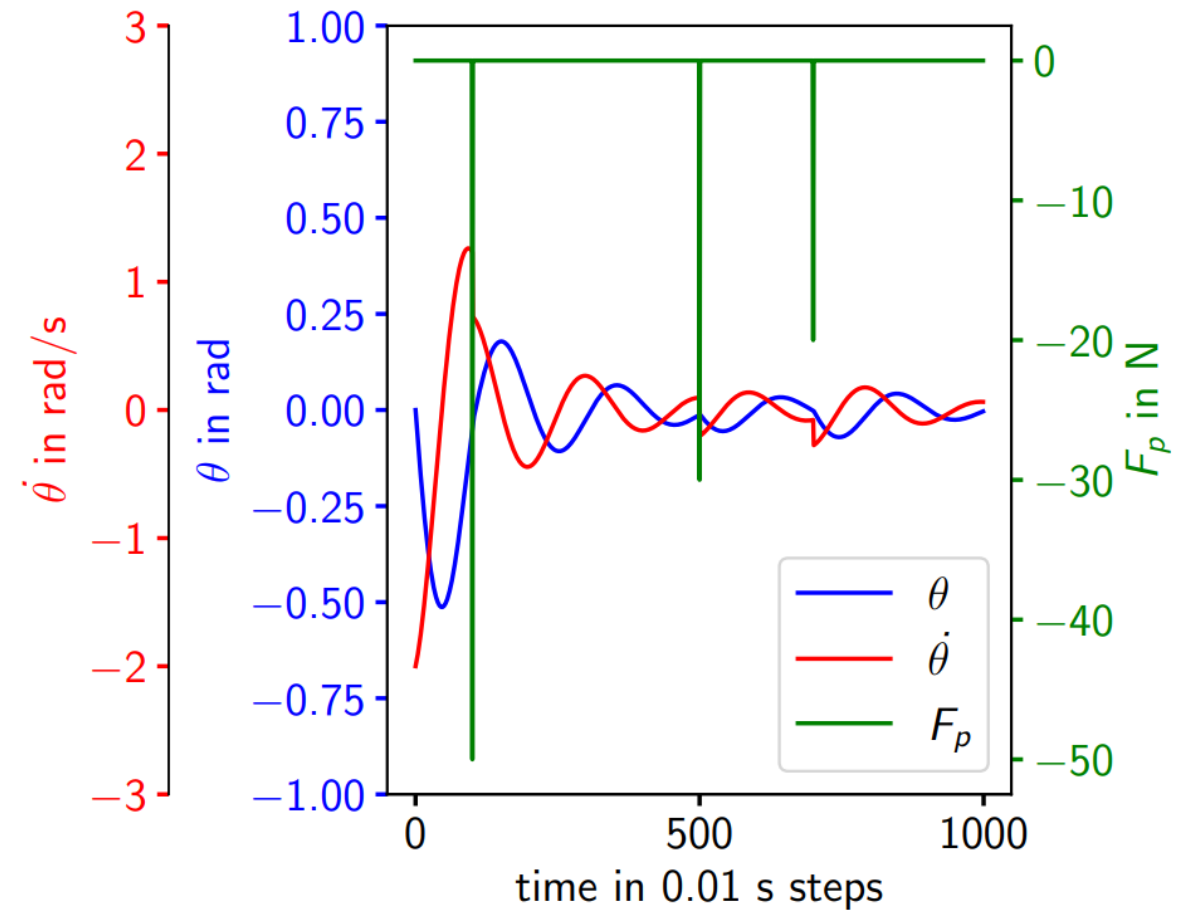
- Example of model with 32 nodes and 2 hidden layers
- Transfer of states to next time steps
- Same weights at all time steps of individual Cells
- Ability to predict indefinite future states
- Fixed number of timesteps for training



# Dataset Creation

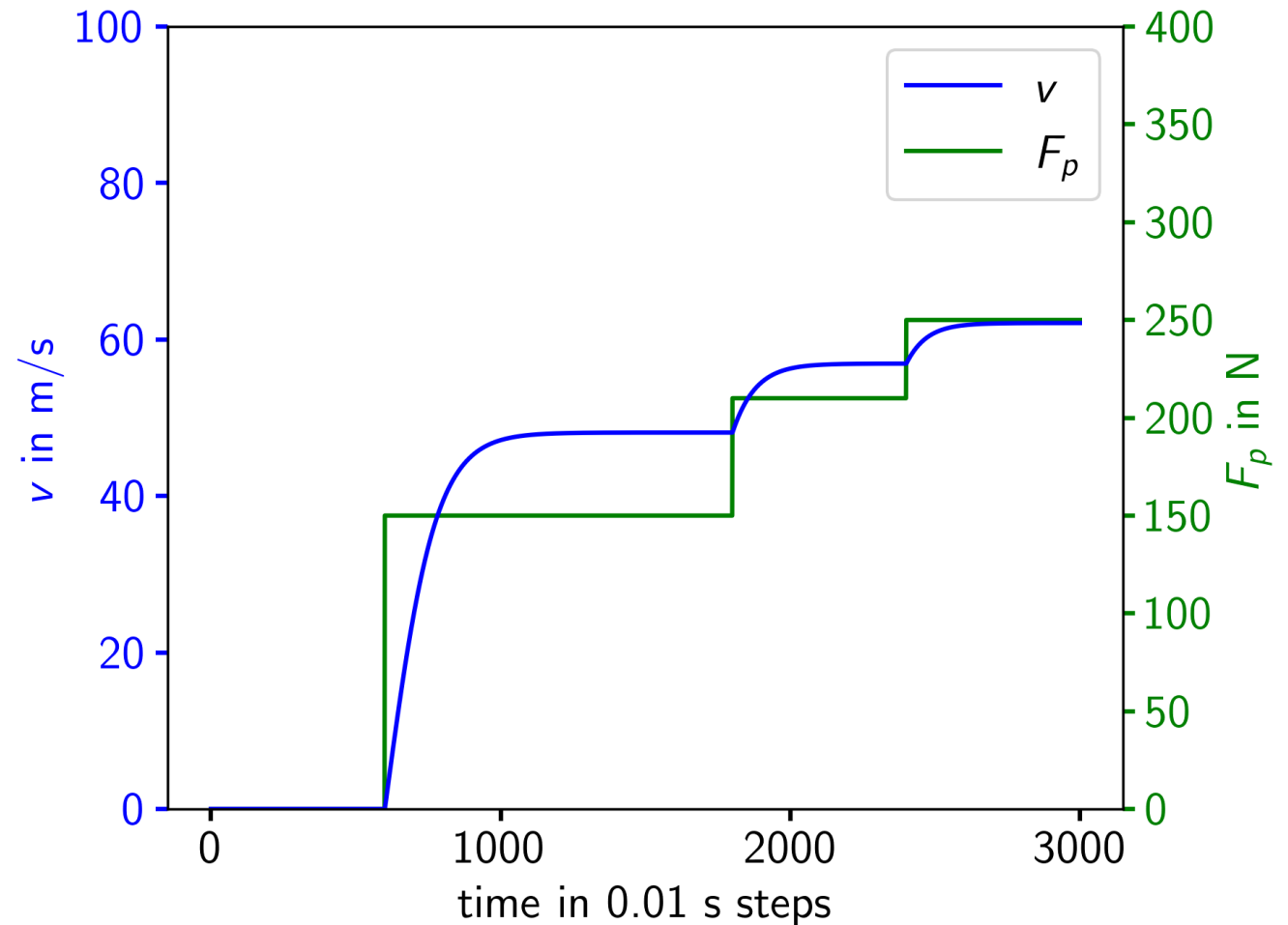
- Pushing a pendulum
- Dampened due to friction
- Varying shape of pushing force  $F_p(t)$
- 200 different time series

$$\frac{d\theta}{dt} = \dot{\theta}$$
$$\frac{d\dot{\theta}}{dt} = \frac{F_p(t)}{ml} \cos(ft) - q\dot{\theta} - \frac{g}{l} \sin \theta$$



- Pushing force on object
- Drag on object
- Varying of shape of pushing force  $F_p(t)$
- 40 different time series

$$\frac{dx}{dt} = v$$
$$\frac{dv}{dt} = F_p(t) - \frac{b}{m} v^2$$





- Split into training, validation and test set
- Scaling of data with MinMaxScaler to  $[-1,1]$  to match LSTM output
- Transformation into correct format
  - Initial condition given at first timestep, zero afterwards
  - Force input given at every time step

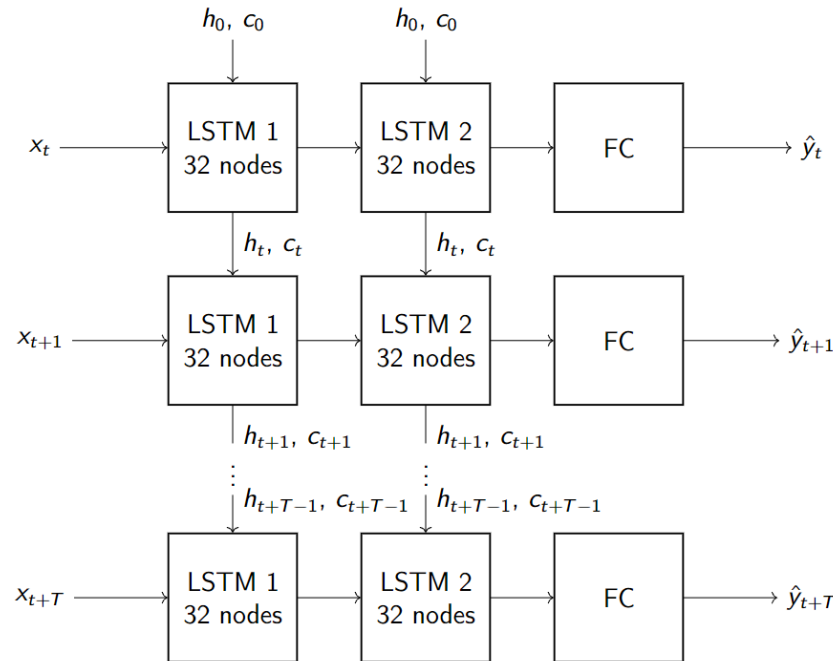
$$X = \begin{bmatrix} \theta_t & \dot{\theta}_t & F_t \\ 0 & 0 & F_{t+1} \\ 0 & 0 & F_{t+2} \\ \vdots & \vdots & \vdots \\ 0 & 0 & F_{T-1} \end{bmatrix}$$

$$y = \begin{bmatrix} \theta_{t+1} & \dot{\theta}_{t+1} \\ \theta_{t+2} & \dot{\theta}_{t+2} \\ \theta_{t+3} & \dot{\theta}_{t+3} \\ \vdots & \vdots \\ \theta_T & \dot{\theta}_T \end{bmatrix}$$

# Putting it together

- Calculation of loss from forward pass
- Update of weights through backpropagation and Adam optimizer

$$X = \begin{bmatrix} \theta_t & \dot{\theta}_t & F_t \\ 0 & 0 & F_{t+1} \\ 0 & 0 & F_{t+2} \\ \vdots & \vdots & \vdots \\ 0 & 0 & F_{T-1} \end{bmatrix}$$



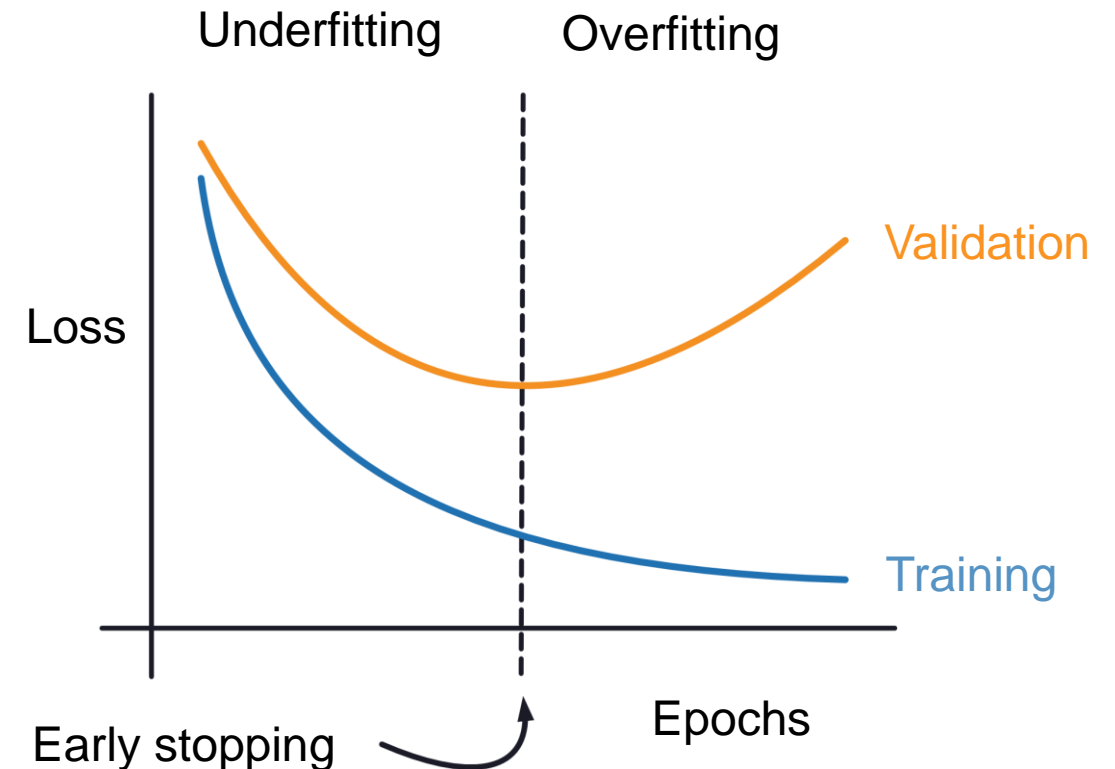
$$y = \begin{bmatrix} \theta_{t+1} & \dot{\theta}_{t+1} \\ \theta_{t+2} & \dot{\theta}_{t+2} \\ \theta_{t+3} & \dot{\theta}_{t+3} \\ \vdots & \vdots \\ \theta_T & \dot{\theta}_T \end{bmatrix}$$

$$\min_W L(X, W, y)$$

$$L = \sum_{t=0}^T (\hat{y}_t - y_t)^2$$

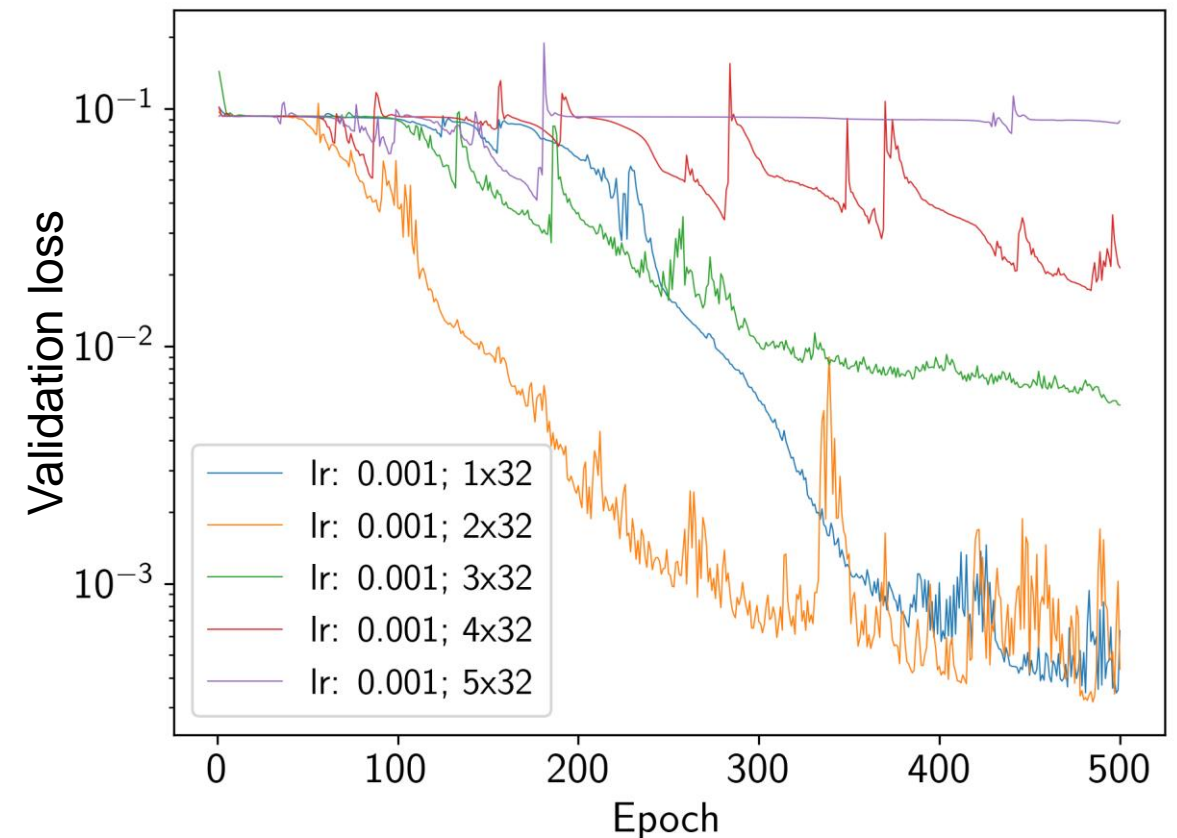
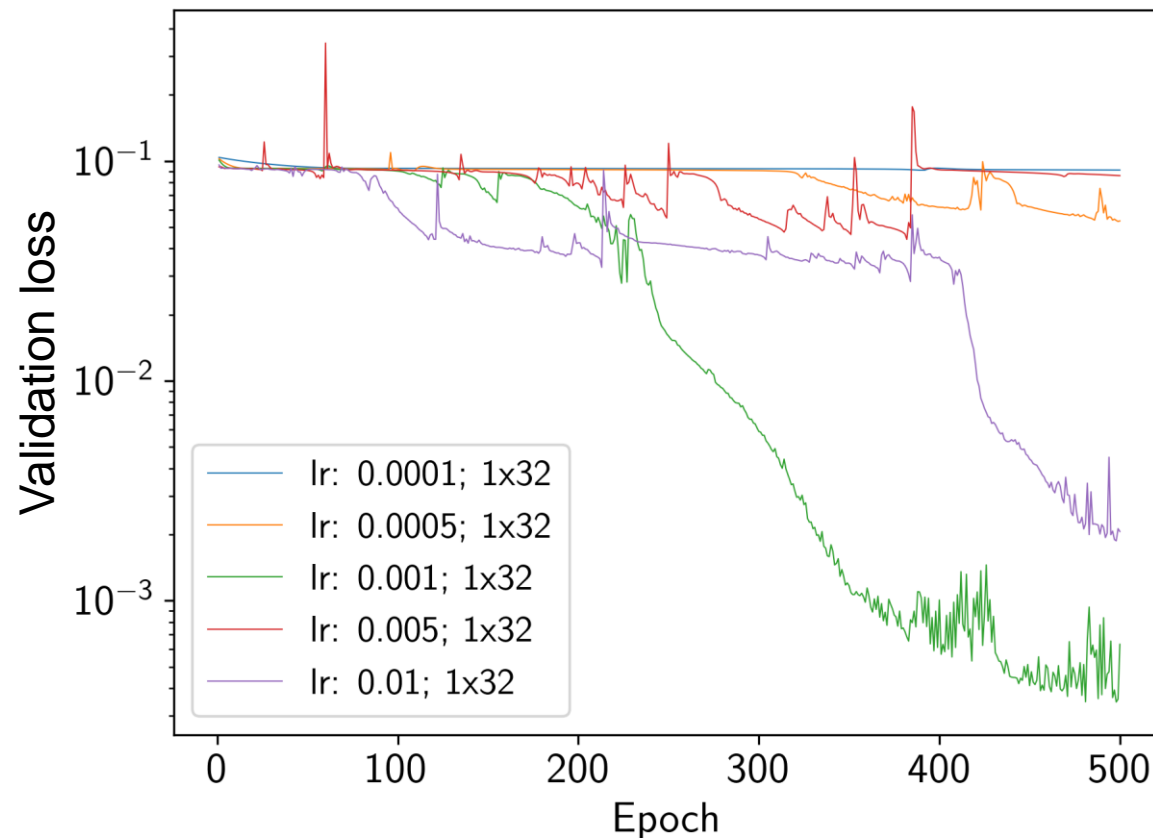
# Hyperparameter Search

- Monitoring validation loss for early stopping
- Check hyperparameters one by one
- Take best parameter for next step
- Search order:
  1. Learning rate
  2. Number of layers
  3. Number of nodes in each layer
- Training for 500 epochs, then 5000 with best model

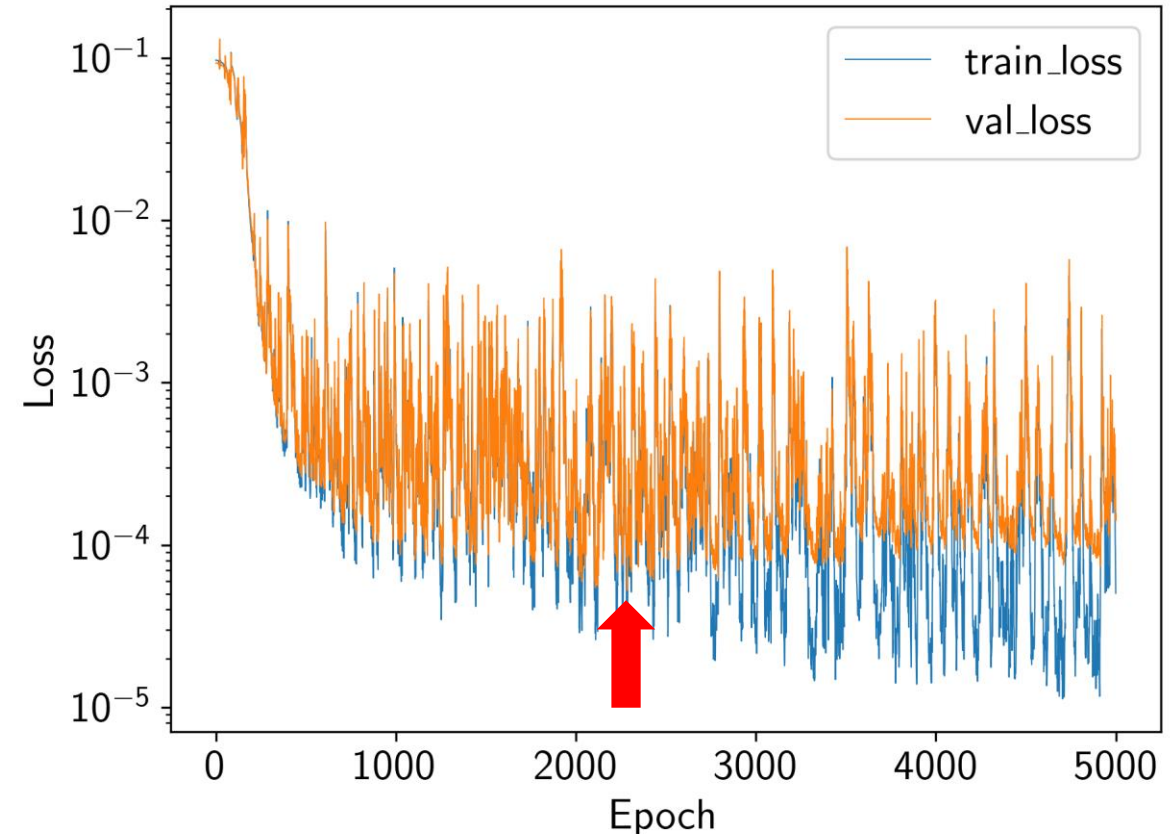
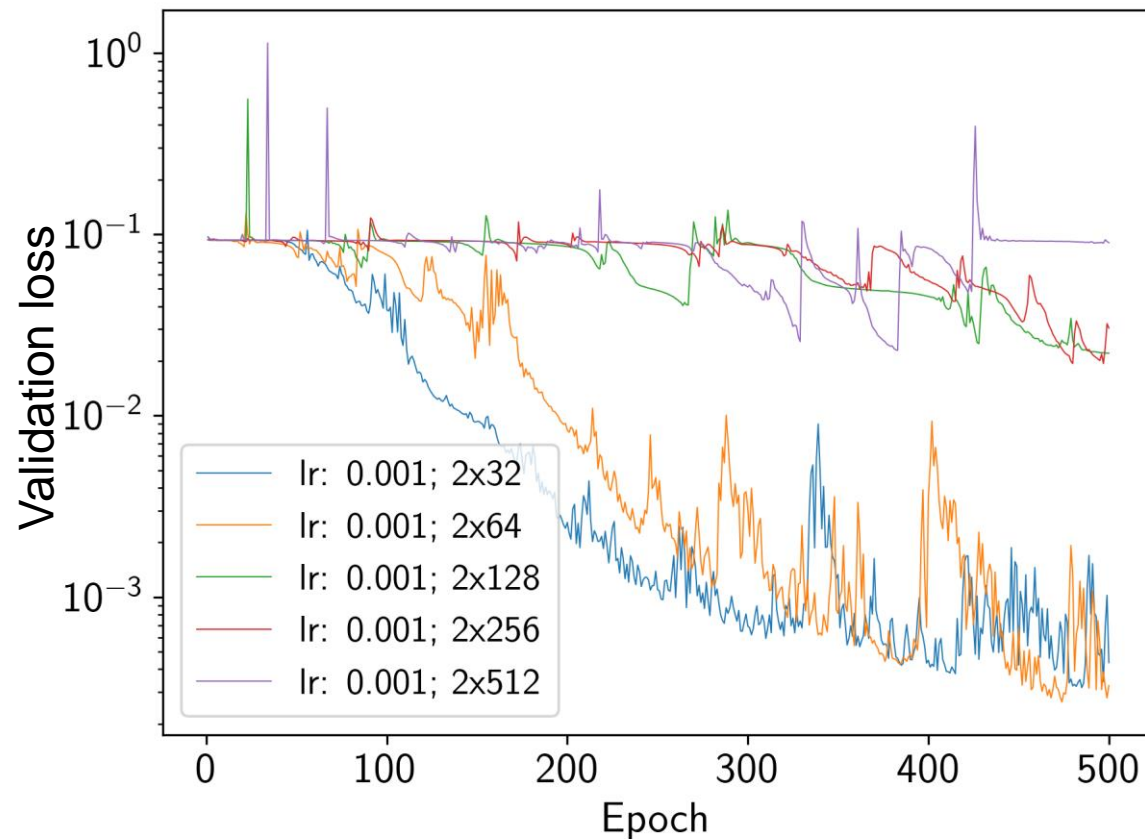


Adapted from (Ryan, H. & Alexis C. 2022)

- Better training performance with higher learning rate and less layers

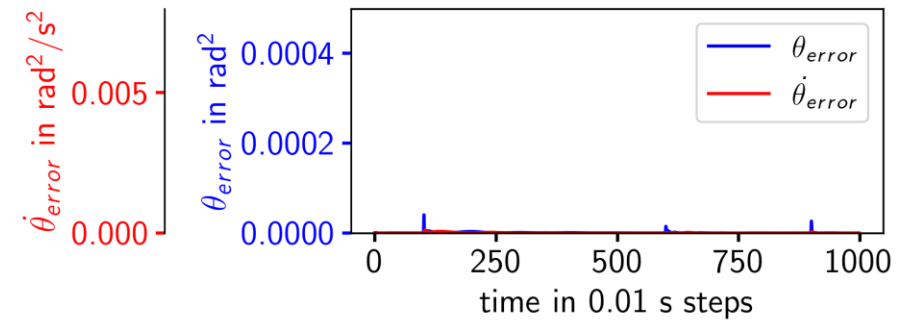
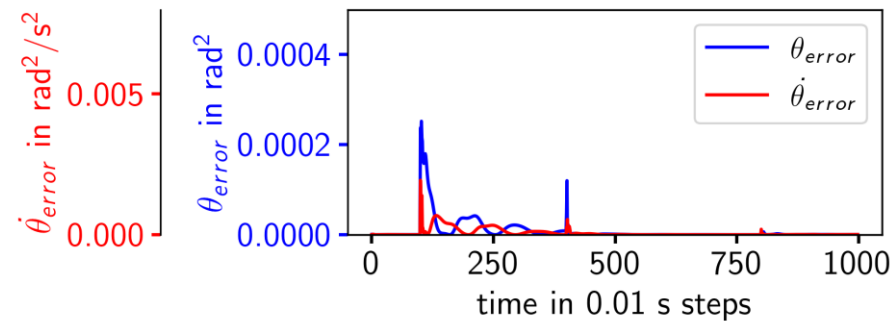
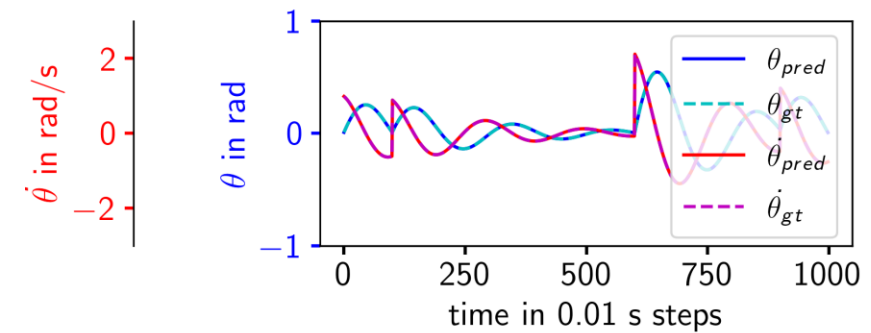
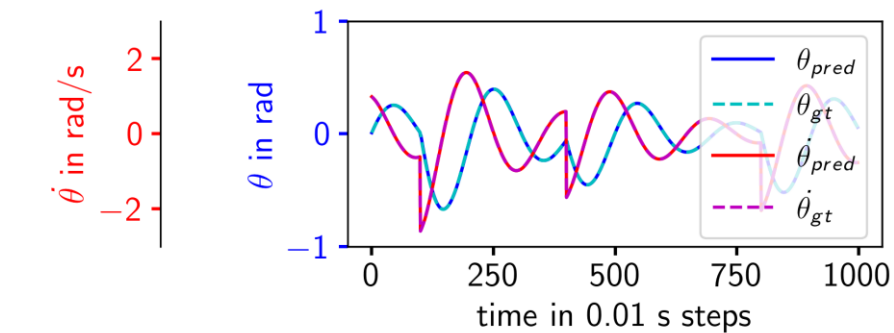


- Higher training stability for smaller models and early stopping to save model with best validation loss



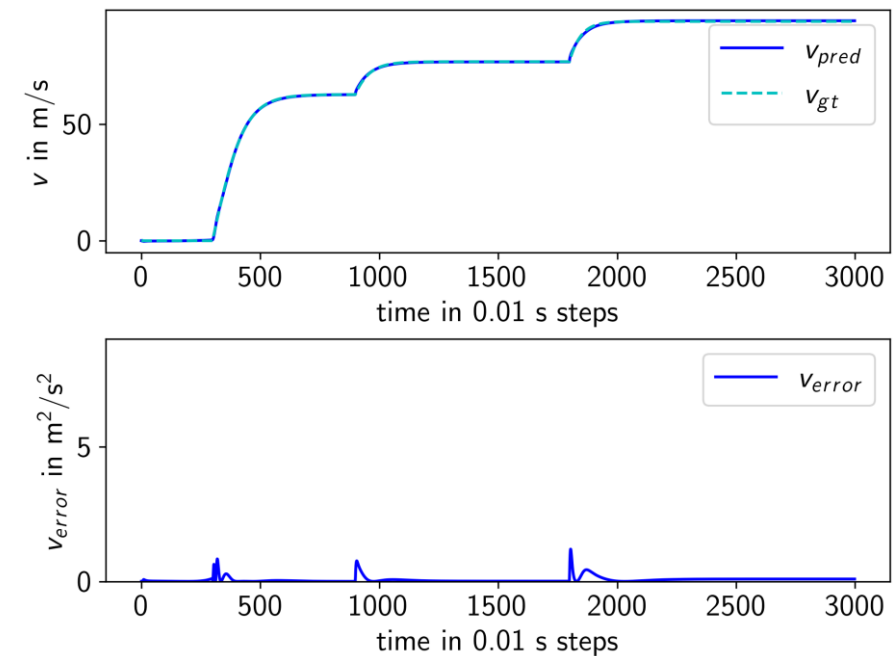
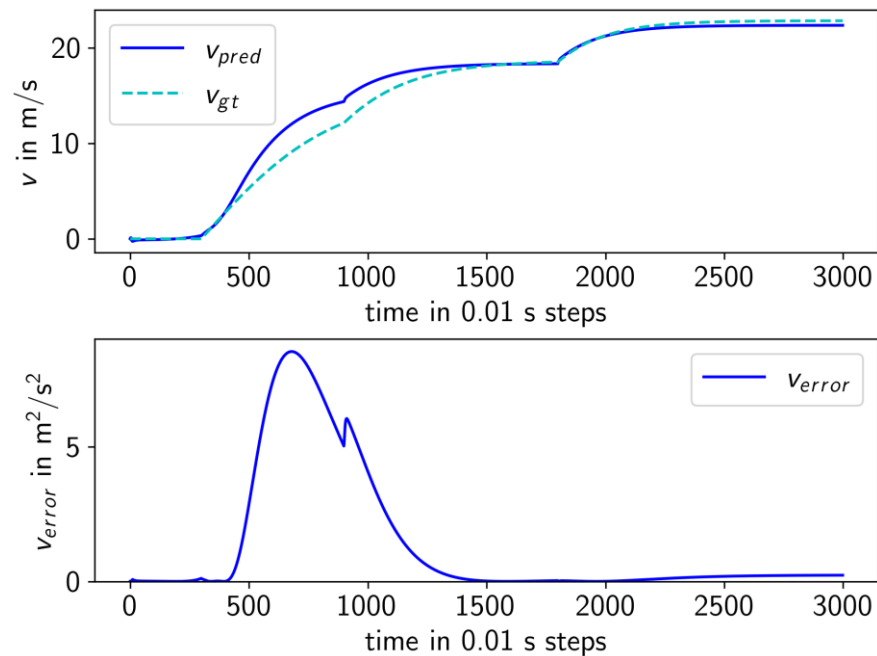
# Evaluation and Conclusion

- Overall low error
- Small error spikes on input change





- Rare large error
- Error spikes on input change



- Conclusion
  - Training difficulties with larger models
  - Error spikes on input change
  - Clear ability to predict nonlinear dynamic system
  - Recommendation:
    - 2 LSTM layers with 32/64 nodes and learning rate of 0.01 for similar systems
    - New hyperparameter search for different/more complex systems
- Future Work
  - Deeper look into training instabilities with larger models necessary
  - Testing for more complex ordinary differential equations
  - Analysis on what and how much training data is necessary to achieve good results

# Thank You!

- Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A review of recurrent neural networks: LSTM cells and network architectures. *Neural Computation*, 31(7), 1235–1270.
- Holbrook, R.; Cook, A. (2022, May 5). Overfitting and underfitting. Kaggle. Retrieved January 12, 2023, from <https://www.kaggle.com/code/ryanholbrook/overfitting-and-underfitting/tutorial>

- Pushing force:
  - $F_p = ma$
- Air resistance (Drag):
  - $F_d = \frac{1}{2}\rho v^2 C_D A$
- Simplification:
  - $b = \frac{1}{2}\rho C_D A$
- Resulting force:
  - $F = F_p - F_d$

- Resulting force:
  - $F = F_p - F_d$
- Express as second order differential equation:
  - $m \frac{d^2 x}{dt^2} = F_p - b \frac{dx}{dt}$
- Transform into system of two first order equations:
  - $\frac{dx}{dt} = v$
  - $\frac{dv}{dt} = \frac{F_p - bv^2}{m}$
  - → use in odeint python function to get values of velocity and/or position