Transformer-based Learning Models of Dynamical Systems for Robotic State Prediction

Alec Reed

alec.reed@colorado.edu

University of Colorado Boulder

Doncey Albin

University of Colorado Boulder

Anuh Pasricha

University of Colorado Boulder

Alessandro Roncone

University of Colorado Boulder

Christoffer Heckman

University of Colorado Boulder

Research Article

Keywords: Robotics, Dynamics Modeling, Deep Learning, Transformer

Posted Date: February 19th, 2024

DOI: https://doi.org/10.21203/rs.3.rs-3919154/v1

Additional Declarations: No competing interests reported.

Transformer-based Learning Models of Dynamical Systems for Robotic State Prediction

Alec Reed, Doncey Albin, Anuj Pasricha, Alessandro Roncone, Christoffer Heckman

Intelligent Robotics Laboratory, Computer Science Department, University of Colorado Boulder.

Contributing authors: firstName.lastname@colorado.edu;

Abstract

In this paper, we propose a novel approach to data-driven dynamical forecasting using transformer-based learning methods. We explore and evaluate the effectiveness of this approach by developing new benchmarks for generalizable, data-driven dynamical forecasting on a robotic arm and a 4-wheeled ground vehicle. The transformer-based approach utilizes a spatio-temporal forecasting model called the *Spacetimeformer* (STF), which was originally designed for long-horizon fields such as weather and economics. We demonstrate that with key innovations, the STF model can also be a powerful tool for short-range dynamical forecasting. The results show that our approach can even outperform popular analytical dynamics models, such as the bicycle model for four-wheeled vehicles and the robot dynamics model for serial manipulator arms.

Keywords: Robotics, Dynamics Modeling, Deep Learning, Transformer

1 INTRODUCTION

Dynamical system modeling is a crucial tool for understanding and predicting the behavior of complex systems in the physical world. Accurate models enable us to optimize the performance of engineering systems, prevent failures and increase the speed of system operation. Despite advances in data-driven forecasting, accurate predictions of complex dynamical systems remain a challenging task. Traditional analytical models often require assumptions about the underlying physics and external forces in the system, making them prone to errors in environments where those assumptions are challenged. In contrast, machine learning-based approaches have shown great promise in learning complex relationships from data, but they can struggle to meet the accuracy of a well tuned analytical model. However, as transformer-based networks continue to outperform previous benchmarks in common learning tasks such as natural language processing [1, 2] and computer vision [3, 4], we see an opportunity to leverage these models to improve the fidelity of learned dynamics models in robotics. In this paper, we propose a novel approach that utilizes the cutting-edge capabilities of transformer-based networks to achieve accurate and generalizable dynamical forecasting.

An ideal data-driven dynamics model should be both highly accurate for the given system and simply generalizable to allow for simple application to various dynamical systems. The most apparent metric of any dynamics model is high accuracy single step state prediction. Popular

localization methods such as the Kalman filter [5] or the particle filter [6] often include dynamics models to ground sensor pose estimates for enhanced performance. Additionally popular path following and trajectory tracking methods such as MPC [7] and MPPI [8] use dynamics models as a basis for future pose estimation when computing the optimal control(s) to execute. Increasing the accuracy of these underlying dynamics models only improves the performance of existing algorithms.

While high-accuracy models are important for the model to be usable, generalizability to a variety of systems makes a data-driven dynamics model truly invaluable. The transformer-based model we propose in this paper is shown to be highly accurate and generalizable between two dissimilar systems.

In general, transformers provide additional benefits when compared to other deep learning models. Specifically related to dynamics modeling, a transformer-based method has the following advantages:

1. **Multi-Step State Forecasting**: When compared to feed forward neural networks (FFNN's) and analytical models, transformers provide the ability to perform time series forecasting, or extended trajectory forecasting. For this reason, LSTMs have also began to gain popularity in this field [9, 10].
2. **Attention-Based Learning**: The enhanced usage of attention is what makes the transformer such a popular and effective neural network model. In addition to quantitative performance gains, attention activations allow the user to see where the model is focusing in its predictions, building confidence that the model is making the correct connections between features. This attribute is shown dramatically by [3], where transformers are applied to image classification problems. The attention mechanism can be used as a "sanity check", to *highlight* the states where the transformer is focusing to make a classification decision.

2 Related Work

The problem of dynamics modeling can be defined follows. Let $X \subseteq \mathbb{R}^n$ and $U \subseteq \mathbb{R}^m$ be the state space and control space, respectively, of a robotic

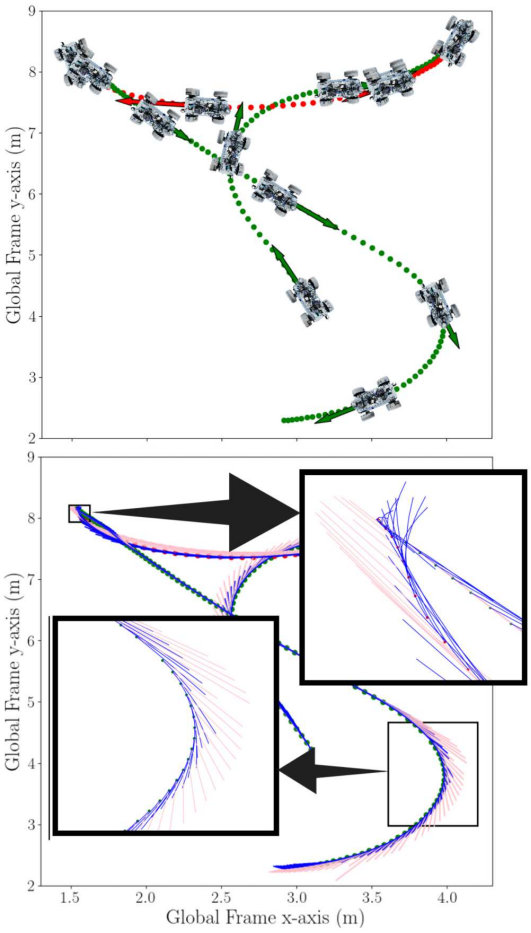


Fig. 1: (Top) A subset of the ground truth path for the Hunter SE 4-wheeled vehicle experiment. Green arrows and dots indicate positive vehicle velocity, while red arrows and dots indicate negative vehicle velocity. (Bottom) A comparison of forecasted trajectories from the STF and the bicycle model [11] on a subset of the testing trajectories. The STF model's predicted trajectories are shown in blue, and the bicycle model's predictions are in red.

system, and let us assume the dynamics of the robot are given by

$$\dot{x}(t) = f(x(t), u(t)), \qquad x(t) \in X, u(t) \in U. \quad (1)$$

Then the problem of dynamical modeling is simply to find some f that most accurately generates $\dot{x}(t)$ given $x(t)$ and $u(t)$.

As discussed in [12, 13] there are four primary methods for system dynamics modeling; analytical models, data-driven models, simulator-augmented residual models and recurrent simulator-augmented residual models. In this work we implement data driven models as a direct replacement for the analytical model. The primary purpose of this approach is to ensure the model is generalizable to any system. To implement any of the other models an analytical or physics model of the system is required.

Analytical dynamics models are a method for approximately solving the dynamics of a system. Popular systems such as 4-wheeled vehicles [11, 14], robotic arms [15], and multi-rotors [16] all have analytical dynamical models that are widely used. However, these models are always designed with underlying physical assumptions in mind, and can perform poorly when adapted to real-world scenarios.

Data-driven dynamics models aim to solve this issue by developing a model directly from the target environment. The idea is that by using data that is representative of real-world system dynamics, the models learn to account for external effects such as slip, friction or imperfect state estimation. FFNNs are one commonly-used technique for data-driven dynamics models. There are examples of successful applications of neural networks for dynamical systems such as vehicles [17], ships [18], multi-rotors [19], and soft robotics [20]. While some of the these systems are data-driven dynamical models due to complex non-linear dynamics, many simply seek a better method than the traditional analytical models.

An alternative to FFNNs for dynamics modeling have been techniques that maintain histories of the state vector. One example of these are LSTMs, which have been used for longer range trajectory prediction, such as human trajectory prediction [21] and vehicle trajectory prediction [22]. Another such technique, transformers [1], have been shown to at times outperform LSTMs in these complex trajectory prediction problems [21, 23]. Unlike LSTMs which process sequences of features consecutively, transformer networks use a self-attention mechanism to process sequences of inputs all at once. The ability to review sequences as a complete dataset rather than one at a time is what drives the favorable comparative performance of transformers when compared to LSTMs.

Remarkably, Eq. (1) represents dynamics which are explicitly dependent only on the current time-step t, whereas these machine learning approaches for trajectory prediction consume states leading up to those at the current time-step. In sequence these states provide a basis for forward-prediction of dynamics using a learned dictionary of motion primitives, such as those frequently used in path planning (e.g., [24]). However there are two advantages to the use of machine learning primitives: 1) they are explicitly learned from data, thus may more accurately capture difficult-to-model effects of robotic platforms, and 2) they are generally highly parallelizable due to their arithmetic structure when placed into the appropriate computing substrate such as general-purpose graphics processing units.

3 Preliminaries and Methods

In this paper we aim to generate a data-driven dynamics model from a time series transformer network. We define 2 models for discussion, the time series transformer (TST) and the spacetime-former (STF) [25]

3.1 Time Series Transformer (TST)

The (TST) [26] is an adaptation of [1]. An example TST implementation is provided by [26] which uses the TST model to forecast influenza prevalence among a sample population. This model relies of time-sequenced vectors of data as inputs to the transformer model, and produced vectors of the same dimensional as outputs as time-sequenced data forecasts. The detailed processing and outputs over the data vectors are displayed in Figure 2.

In [26], a single quantity of interest was being predicted (influenza prevalence), only from previous measurements of that quantity. Notably, it is seen that increasing the dimensionality of the feature vectors (by adding constructed features the authors thought to be informative) does not clearly improve the performance of the model. As analyzed in [25], this is a common outcome when attempting to provide multivariate feature vectors to TSTs. Crucially for these techniques, attention is only computed between *vectors* of features, not the features themselves. Since the additional features cannot be directly attended

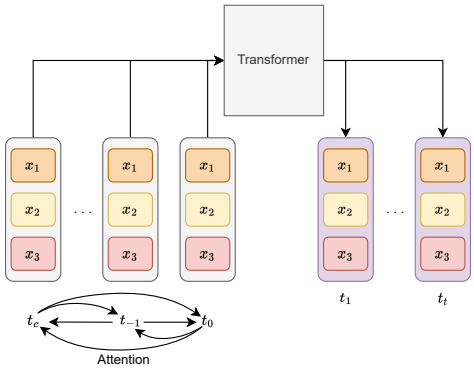


Fig. 2: An example of the input and output feature vectors for a TST. Self-attention weights are shown as black lines for the feature vector inputs.

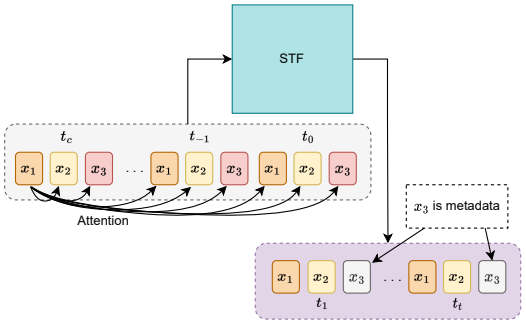


Fig. 3: An example of the input and output feature vectors for the STF model omitted metadata. Note the same feature vectors from Figure 2 are flattened, allowing self attention to be calculated between every feature. Example self attention weights for feature x_1 at time t_c are shown as black lines for the feature vector inputs, however there are attention weights computed between every input feature at every time step.

to while they exist in vector from, adding these features commonly does not increase performance.

Given the high dimensionality of robotics datasets, a TST is not a generally effective method for data-driven dynamics modeling.

3.2 Spacetimeformer Model (STF)

To address the challenge posed by TSTs not attending across feature vector elements, the STF model allows for attention weights to be computed between each feature in a vector. While at its core the STF is a traditional encoder/decoder transformer network, this modification allows it to substantially improve forecasting performance when the dimensionality of the feature vectors is larger than 1. To allow for this per-feature attention model, the feature vector is flattened before providing values to the model as shown in Figure 3. The features are still related in time by temporal attention and positional encodings, however now the features can also be related by spatial attention. While computationally expensive, this method offers vast improvement over standard TSTs for time-series predictions on multi-dimension data.

In the field of robotics there is often additional data beyond the desired state outputs that can provide context to the system models. These data are frequently referred to as *metadata*. To further enhance the performance of the STF model, it is desirable to provide metadata as additional input features. However in the classical STF implementation, these additional features are forecasted as well; this can lead to decreased performance for inference over the features of interest due to additional normalization and training requirements. To focus the model only on the features of interest, we use a custom loss function defined in Eq. (2):

$$l = \frac{1}{n}\sum(x_{gtr} - x_{fr})^2, \tag{2}$$

where x_{gtr} is the relevant ground truth features, x_{fr} are the relevant forecasted features, and $n = \dim x_{fr}$.

The loss function defined in Eq. (2) is simply the mean squared error, however it ignores the forecasted metadata when computing the training loss. We show in Section 4 that this modification is highly effective at increasing the accuracy of forecasted states.

4 Experiments and Results

In this section we show the results of training the STF model on two dissimilar systems. For each a popular analytical model is used as a point of comparison, as well as a LSTM model and a FFNN model.

4.1 4-wheel Vehicle

This experiment explores generating a dynamics model for a real world Hunter SE 4-wheeled

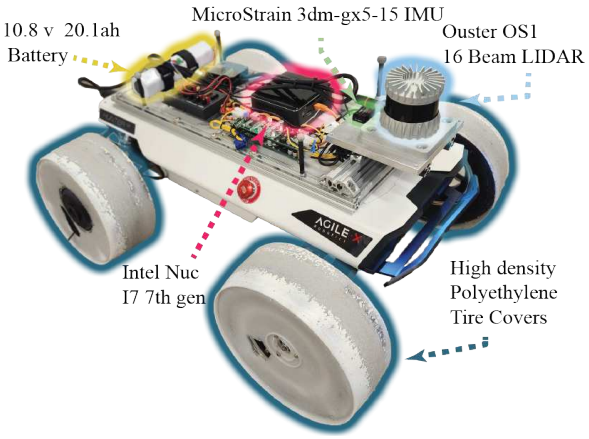
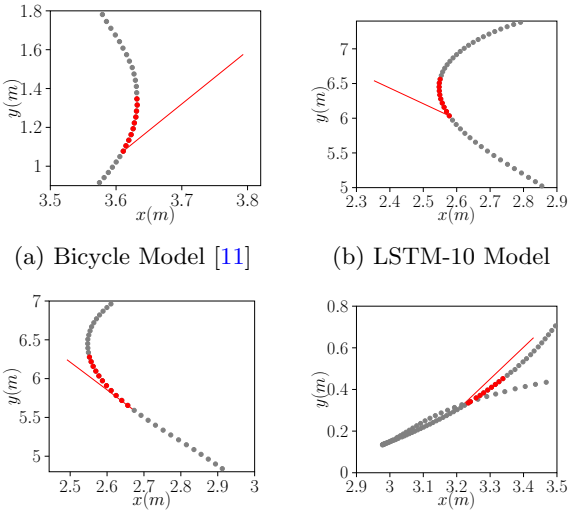


Fig. 4: Hunter SE platform with labeled perception and compute components.



(a) Bicycle Model [11]

(b) LSTM-10 Model

(c) STF

(d) FF Model [17]

Fig. 5: Select P90 results showing large translation errors for each model. Gray points are measured states, while red dots are the ground truth future states. Red lines are the associated models forecasted trajectory. (a) When performing an aggressive left turn, the bicycle model is unable to accurately project the future state due to the noise in the velocity measurements and the large dt the model is attempting to predict. (b) The LSTM-10 model does very well projecting the vehicle in gentle maneuvers but struggles to correctly predict the motion of the vehicle in an aggressive sliding turn. (c) The max STF translation error is on a straight away just about to turn. Since the model has no context that the turn is coming, it predicts the path will continue straight. (d) the FF model struggles to accurately forecast future states after an aggressive maneuver. The gray dots here indicate that the vehicle was in reverse and then changed direction quickly, inducing noise and slip into the vehicle state.

ground vehicle by Agile-X as shown in Figure 4. The system is retrofitted with plastic tires to reduce the the friction between the vehicle and the ground, making the dynamics less predictable by increasing vehicle slip. As a result the vehicle slides when executing sharp turns or hard stops. The experiment was conducted in a motion capture space that provides ground truth pose estimates based on fiducial's located on the vehicle. To collect the necessary data to train and test the network, we teleoperated the platform for approximately four minutes. The collected dataset consisted of 100 Hz vehicle pose updates from the motion capture system and 50 Hz teleoperator input commands (for velocity and steering). To generate our full dataset, we reduced the number of pose estimates by half, resulting in approximately 10,000 data samples. While this is a fairly small dataset, it shows how quickly the data-driven models can learn from scratch. The goal of each dynamics model is to estimate the future state of the vehicle 0.2 s in the future.

4.1.1 Experiment

For this experiment we compare the STF model against a traditional bicycle model [27], 2 versions of LSTM's, and the feed-forward network as described in the popular Georgia Tech rally car experiment [17]. During experimentation, we found that the P50 performance of the LSTM could be improved by decreasing the 64 data point context window and removing the control data (u_v and u_δ). For completeness we include results for both the highest performing LSTM (LSTM–10) and the LSTM using the exact same input data as the STF model (LSTM–64). The data is used to predict the future state of each model 0.2 s in the future. Here, we discuss the inputs and outputs for each model being compared. $\dot{x}, \dot{y}$ define the change in x and y position of the vehicle respectively. The quaternion that defines the rotation of the vehicle

is defined as the states q_x, q_y, q_z, q_w. The control throttle is represented as u_v, which is a value [0,1]. The control steering angle is represented as u_δ and is a value [-1,1] :

- *STF*: 64 data points consisting of $\{\dot{x}, \dot{y}, q_x, q_y, q_z, q_w, u_v, u_\delta\}$ collected at a frequency of 50 Hz are used context points to predict the next 10 data points at 50 Hz. While all input values are forecasted the loss function only calculates loss on $\{\dot{x}, \dot{y}, q_x, q_y, q_z, q_w, \}$.
- *LSTM–10*: The LSTM model used 50 hidden states to process 10 input data points consisting of $\{\dot{x}, \dot{y}, q_x, q_y, q_z, q_w, \}$. A linear layer is added to the output of the LSTM to process the output of the hidden states and predict the next 10 target points.
- *LSTM–64*: The same model as the *LSTM–10* but uses the same number of context points (64) and the same states as the *STF* model for a direct comparison. This model also predicts the next 10 target points.
- *Feedforward Network (FF)*: The current time step consisting of $\{\dot{x}, \dot{y}, \dot{\theta}, u_v, u_\delta\}$ is used to predict the next time step of $\{\dot{x}, \dot{y}, \dot{\theta}\}$ as described in the Georgia Tech rally car experiment [17].
- *Bicycle Model*: The bicycle model [11] is a very popular analytical model for estimating the dynamics of a 4-wheeled vehicle. A popular simplified version of this model is computes state derivatives as follows:

$$\begin{aligned} \dot{x} &= v\cos(\theta) \\ \dot{y} &= v\sin(\theta) \\ \dot{\theta} &= \frac{\tan(\delta)}{L}. \end{aligned} \tag{3}$$

  Where θ defines the yaw angle of the vehicle, δ defines the steering angle of the vehicle and L is length between the front and back axle.

4.1.2 Results

Table 1 shows the results for each model when applied to the test set. In terms of translation the STF model reduced the P50 translation drift by 7.5% when compared to the LSTM-10 implementation and 36% when compared to the bicycle model. However the STF model most outperforms other implementations in the precision of the estimates, where the max translation drift is reduced 48% and 55% when compared to the LSTM-10 and bicycle model respectively. Figure 5 shows a trajectory with high relative error from each model. While each model exhibit larger than average error when forecasting during aggressive maneuvers the STF models max error is displayed when transitioning from a long straight trajectory to an aggressive turn as shown in Figure 5c. Since the model has no context points indicating the turn is about to start, it forecasts as if it is going in a straight line.

The STF model struggles to outperform the bicycle model, both LSTM's and even the FF network in heading drift. While these results are surprising, they are not completely unique as deep learning models have been known to struggle with rotational data and predictions. Various rotational encodings such as rotational matrices and quaternions were evaluated, however the STF performance in heading drift remained low compared to the competitors. It is likely that the network is not able to correctly calculate attention between the rotational data and translation. This problem is an open field of research and some progress has been made to generate network friendly rotational mapping to increase the performance of the network [28].

Finally, the STF model's ability to forecast more complex trajectories is shown particularly well in Figure 1. As shown by the right-most cutout box, where the vehicle is sliding to a stop before reversing direction, the bicycle model is only able to predict the continuing forward motion of the vehicle. The STF model is able to provide the additional insight that the vehicle is about to change direction, as show by the blue trajectories beginning to "wrap" around just before the change in direction. This additional forecasting capability can be incredibly valuable when planning motion around obstacles or generating safety boundaries of a system.

4.2 7DoF Robotic Manipulator

4.2.1 Platform and Data Collection

To explore the generalizablity of these models we train the same models on a real world seven degree-of-freedom (DoF) Franka Emika Panda robot arm. This serially-linked kinematic chain is operated by a human in gravity compensation mode. The operator randomly moves arm within

Table 1: Model results for Hunter vehicle. Here dt is equal to the future prediction time, 0.2 s. T – Drift stands for translation drift of the vehicle and θ – drift stands for the heading drift of the vehicle

	Translation Drift (cm/0.2 sec)			Heading Drift (rad/0.2 sec)		
Model	P50	P90	Max	P50	P90	Max
STF [25]	**3.705**	**7.109**	**13.401**	0.119	0.256	0.321
Bike [11]	5.781	12.10	30.02	**0.031**	**0.070**	**0.175**
LSTM–10	4.007	15.219	26.062	0.059	0.224	0.747
LSTM–64	4.341	9.047	19.731	0.063	0.226	0.430
FF [17]	6.316	12.02	25.14	0.121	0.204	0.339

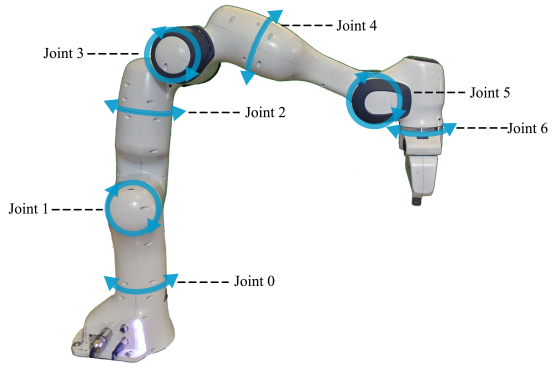


Fig. 6: The 7DoF Franka Emika Panda robot arm.

its reachable workspace, ensuring complete coverage of joint angle, velocity, and acceleration ranges for each of the seven joints. A trajectory of approximately 10 minutes is recorded at a rate of 1 kHz.

Each data point is a set $\{q, \dot{q}, \tau, M, C, g, \tau_{ext}, J\}$, where q is a 7×1 vector representing the robot joint angles, $\dot{q}$ is joint velocity 7×1 vector, τ is a 7×1 vector defining joint torques, M is a positive-definite, symmetric 7×7 joint inertia matrix, C is a 7×7 matrix that represents the Coriolis and centrifugal effects of robot motion, g is a 7×1 vector that describes how much torque each motor needs to apply to oppose gravity, τ_{ext} is a 7×1 vector describing external joint forces, and J is a 7×7 matrix defining the Jacobian of the robot arm at a given configuration.

4.2.2 Baseline Model

The STF model is also compared against an analytically-derived dynamics model for the Franka Emika Panda arm [15]. This analytical model is defined as:

$$\tau(q) = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) + f(\dot{q}), \quad (4)$$

where τ, q, $\dot{q}$, M, C, and g are defined as before, and f is a 7×1 vector that denotes the torques needed to overcome joint friction.

4.2.3 Experiment

We compare the STF model against a LSTM and FFNN. For maximum performance 2 models are trained for each deep learning model. One model to prediction the q states and another model to predict the $\dot{q}$ states. Additionally we compare our results against a sate of the art analytical model [15]. Similar to the 4-wheeled vehicle experiment we found that we could increase the performance of the LSTM model by reducing the context window and removing the τ_n data. We include both the LSTM trained on the same inputs as the STF model (LSTM–64) as well as the highest performing LSTM model (LSTM–3) for completeness. Each model was used to predict the future state of the system 0.1 s in the future. Since the data capture frequency of the Panda arm is 1kHz and predictions are made at 10Hz, control input τ is applied at $t = 0, 0.1, \ldots, T - 0.1$ to the arm dynamics model and set to 0 in all the intermediate timesteps (e.g., in range $(0, 0.1), (0.1, 0.2)$, etc.) during the prediction process. Integration of the arm dynamics model (Eq. 4) is performed using numerical integration.

Table 2: Results for the robot arm position (q) and velocity ($\dot{q}$) estimates. Prediction is in rads/dt where dt is 0.1 seconds. $\bar{q}$ and $\bar{\dot{q}}$ represent average join position error and average joint velocity error respectively

		q_0	q_1	q_2	q_3	q_4	q_5	q_6	$\bar{q}$	$\dot{q}_0$	$\dot{q}_1$	$\dot{q}_2$	$\dot{q}_3$	$\dot{q}_4$	$\dot{q}_5$	$\dot{q}_6$	$\bar{\dot{q}}$
P50	STF [25]	0.045	0.038	0.029	0.055	0.024	**0.014**	0.0059	**0.0301**	**0.072**	**0.086**	**0.077**	**0.095**	**0.096**	**0.054**	0.036	**0.0737**
	GAM [15]	**0.011**	0.0832	**0.024**	0.101	**0.013**	0.053	**0.0020**	0.0410	0.195	1.625	0.444	1.891	0.145	0.822	**0.0211**	0.734
	LSTM–3	0.069	**0.030**	0.137	0.080	0.107	0.029	0.187	0.0912	0.179	0.144	0.181	0.212	0.309	0.184	0.109	0.188
	LSTM–64	0.144	0.092	0.128	**0.046**	0.163	0.124	0.360	0.151	0.137	0.128	0.096	0.147	0.174	0.098	0.107	0.126
	FF [17]	1.831	1.007	3.139	0.697	1.226	0.696	0.9009	1.3567	0.336	0.407	0.273	0.580	0.292	0.0959	0.056	0.291
P90	STF [25]	0.163	**0.095**	**0.081**	0.132	0.135	0.091	0.041	0.105	**0.195**	**0.206**	**0.199**	**0.237**	**0.372**	**0.230**	**0.168**	**0.229**
	GAM [15]	**0.042**	0.135	0.082	0.174	**0.075**	0.163	**0.031**	**0.100**	0.755	2.641	1.686	3.287	0.920	3.029	0.317	1.805
	LSTM–3	0.160	0.100	0.284	0.166	0.268	**0.081**	0.319	0.196	0.561	0.402	0.430	0.637	0.738	0.440	0.427	0.439
	LSTM–64	0.417	0.207	0.261	**0.121**	0.507	0.206	0.476	0.313	0.418	0.404	0.297	0.404	0.556	0.289	0.349	0.388
	FF [17]	3.080	2.130	4.113	1.245	1.787	1.220	1.360	2.133	1.035	0.906	0.884	1.364	1.461	0.707	0.537	0.984
Max	STF [25]	0.315	**0.153**	**0.151**	0.273	0.319	**0.206**	0.190	0.229	**0.517**	**0.411**	**0.546**	**0.511**	**0.949**	**0.626**	**0.902**	**0.637**
	GAM [15]	**0.159**	0.172	0.214	0.235	**0.223**	0.375	**0.173**	**0.221**	3.640	3.248	4.616	4.344	2.863	1.699	1.699	3.158
	LSTM–3	0.327	0.267	0.487	0.255	0.347	0.182	0.423	0.326	1.415	1.070	1.191	1.031	2.057	1.801	1.625	1.455
	LSTM–64	0.725	0.311	0.442	**0.204**	0.661	0.301	0.618	0.466	1.006	0.812	1.202	0.969	1.775	1.592	1.344	1.242
	FF [17]	4.537	2.468	4.291	2.068	2.871	1.442	1.629	2.758	1.895	1.664	1.533	2.557	3.345	2.072	1.642	2.101

- *STF*: 64 data points consisting of $\{q_{0-6}, \dot{q}_{0-6}, \tau_{0-6}\}$ are used as inputs. Data is collected at a frequency of 30 Hz and used as context points to predict the next three data points at 30 Hz (0.1 s in the future).The q prediction model only calculates loss on $\{q_{0-6}\}$ and $\dot{q}$ model only calculates loss on $\{\dot{q}_{0-6}\}$.
- *LSTM–3*: The same LSTM structure that was used in the 4-wheeled vehicle experiment is used here. 3 data points consisting of $\{q_{0-6}, \dot{q}_{0-6}\}$ collected at a frequency of 30 Hz is used as context points to predict the next three data points at 30 Hz (0.1 s in the future). 2 models are trained one that predicts $\{q\}$ and one that predicts $\dot{q}$.
- *LSTM–64*: The same model as *LSTM–3* but using the same inputs as the STF model.
- *Feedforward Network (FF)*: The current time step consisting of $\{q_{0-6}\}$ is used to predict the next time step of $\{q_{0-6}\}$ for the q prediction model and the current time step consisting of $\{\dot{q}_{0-6}\}$ is used to predict the next time step of $\{\dot{q}_{0-6}\}$ for the $\dot{q}$ model.
- *Gaz et al. [15] Analytical Model (GAM)*: An analytical model for predicting the motion of the Franka Emika Panda robot arm. Refer to Section 4.2.2 for discussion.

4.2.4 Results

Unlike the 4-wheeled vehicle experiment discussed in Section 4.1 the STF model learns to predict the angular position and velocities of the robotic arm better than the competing methods. A factor in this relative performance increase may be attribute to the speed at which the joint angle of the robotic arm system change. Simple models may struggle to characterize this behavior while while the more complex STF model is more successful by comparison. Similar to the previous experiment the LSTM-3 and LSTM-64 models produce respectable results. However, the transformer-based method reduces the P50 average error by 70% and 60% for the best performing LSTM model when comparing the joint and joint velocities respectively.

The STF method is competitive with the analytical model on this dataset. In the q estimates, GAM [15] and STF perform similarly, each model performing better than the other on some joint predictions. Notably, STF outperforms GAM is in predicting joint velocities, outperforming the analytical model in every joint at every percentile but P50 j_6. Due to the relatively large time step for the prediction, the velocity estimates for the analytical model are highly effected by the noise inherent in the state measurements. The data-driven model is much better at filtering out the noise naturally and making a reasonable prediction on the future velocities of the robotic arm.

While the LSTM and FFNN do not perform as well as the transformer-based method, they do use far fewer trainable parameters and therefore could be applied in situation where compute is limited. In particular, the LSTM-3 could be used a simple improvement to the FF model that just needs 3 points history to perform very well in both q and $\dot{q}$ predictions. Interestingly the LSTM-3 performs better in q predictions and the LSTM-64 performs better in $\dot{q}$ predictions. Since in general the joint velocity can change much more quickly than the joint positions it is possible that the addition context can aid in velocity predictions, where only a

few context points are required for position prediction. In this experiment, all trained models were able to out perform the analytical model when predicting $\dot{q}$.

5 Conclusion and future work

While analytical models can provide high accuracy estimates for dynamical systems, we show that a well-trained transformer network can outperform even cutting edge analytical methods in some applications. Additionally, while a simpler model such as an LSTM can do well in systems with a smaller or less complex state space, these models struggle to generalize well to more complex systems as shown the increase in error relative to the STF model between the presented experiments. The transformer-based method has been shown to generalize well to dissimilar dynamical systems when predicting both system state velocities and positions.

Future work could include implementing these models in real time to evaluate the trade off between inference time and forecast accuracy. Since the transformer based models are much larger than the other networks the dynamical forecasts would be less frequent. during this real time implementation evaluating the increased dynamical forecast accuracy on end user performance (such as localization or path planning) would provide additional insight to the effectiveness of high accuracy dynamics models. Additionally, the STF model could be used to predict longer range trajectories on the order of seconds. While longer range trajectory prediction is a slightly different field than dynamics modeling, it would be interesting to see how the STF model performs against other current models.

Author Contributions. A.R was the primary author, developed the learning frameworks, generated results, and wrote the main manuscript. D.A assisted with writing the manuscript as well as the 4-wheel vehicle data collection. A.P assisted in writing the manuscript and collecting the data for the robotic arm experiments. All authors reviewed the manuscript.

Conflict of interest. The authors declare no competing interests

Acknowledgment. The authors would like to thank Dusty Woods for his contributions in hardware and figure design.

Funding. This work was supported by Nation Science Foundation (NSF) #1932189.

Data sharing. Not applicable to this article as no datasets were generated or analyzed during the current study

References

[1] Vaswani, A. *et al.* Attention is all you need (2017). URL https://arxiv.org/abs/1706.03762.

[2] Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding (2018). URL https://arxiv.org/abs/1810.04805.

[3] Dosovitskiy, A. *et al.* An image is worth 16x16 words: Transformers for image recognition at scale (2020). URL https://arxiv.org/abs/2010.11929.

[4] Radford, A. *et al.* Learning transferable visual models from natural language supervision (2021). URL https://arxiv.org/abs/2103.00020.

[5] Ribeiro, M. I. Kalman and extended kalman filters: Concept, derivation and properties. *Institute for Systems and Robotics* **43**, 3736–3741 (2004).

[6] Djuric, P. M. *et al.* Particle filtering. *IEEE signal processing magazine* **20**, 19–38 (2003).

[7] Camacho, E. F. & Alba, C. B. *Model predictive control* (Springer science & business media, 2013).

[8] Williams, G., Aldrich, A. & Theodorou, E. A. Model predictive path integral control using covariance variable importance sampling. *ArXiv* **abs/1509.01149** (2015).

[9] Cao, J., Li, Z. & Li, J. Financial time series forecasting model based on ceemdan

and lstm. *Physica A: Statistical mechanics and its applications* **519**, 127–139 (2019).

[10] Sagheer, A. & Kotb, M. Time series forecasting of petroleum production using deep lstm recurrent networks. *Neurocomputing* **323**, 203–213 (2019).

[11] Polack, P., Altché, F., d'Andréa Novel, B. & de La Fortelle, A. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? 812–818 (2017).

[12] Ajay, A. *et al.* Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing (2018). URL https://arxiv.org/abs/1808.03246.

[13] Bear, D. M. *et al.* Physion: Evaluating physical prediction from vision in humans and machines. *arXiv preprint arXiv:2106.08261* (2021).

[14] Pacejka, H. B. & Bakker, E. The magic formula tyre model. *Vehicle system dynamics* **21**, 1–18 (1992).

[15] Gaz, C., Cognetti, M., Oliva, A., Giordano, P. R. & De Luca, A. Dynamic identification of the franka emika panda robot with retrieval of feasible parameters using penalty-based optimization. *IEEE Robotics and Automation Letters* **4**, 4147–4154 (2019).

[16] Heredia, G. *et al.* Control of a multirotor outdoor aerial manipulator 3417–3422 (2014).

[17] Goldfain, B. *et al.* Autorally: An open platform for aggressive autonomous driving. *IEEE Control Systems Magazine* **39**, 26–55 (2019).

[18] Moreira, L. & Guedes Soares, C. Dynamic model of manoeuvrability using recursive neural networks. *Ocean Engineering* **30**, 1669–1697 (2003). URL https://www.sciencedirect.com/science/article/pii/S0029801802001476.

[19] Bansal, S., Akametalu, A. K., Jiang, F. J., Laine, F. & Tomlin, C. J. Learning quadrotor dynamics using neural network for flight control. *2016 IEEE 55th Conference on Decision and Control (CDC)* 4653–4660 (2016).

[20] Gillespie, M. T., Best, C. M., Townsend, E. C., Wingate, D. & Killpack, M. D. Learning nonlinear dynamic models of soft robots for model predictive control with neural networks 39–45 (2018).

[21] Giuliari, F., Hasan, I., Cristani, M. & Galasso, F. Transformer networks for trajectory forecasting 10335–10342 (2021).

[22] Altché, F. & de La Fortelle, A. An lstm network for highway trajectory prediction 353–359 (2017).

[23] Yu, C., Ma, X., Ren, J., Zhao, H. & Yi, S. Spatio-temporal graph transformer networks for pedestrian trajectory prediction 507–523 (2020).

[24] Dharmadhikari, M. *et al.* Motion primitives-based path planning for fast and agile exploration using aerial robots 179–185 (2020).

[25] Grigsby, J., Wang, Z. & Qi, Y. Long-range transformers for dynamic spatiotemporal forecasting (2021). 2109.12218.

[26] Wu, N., Green, B., Ben, X. & O'Banion, S. Deep transformer models for time series forecasting: The influenza prevalence case (2020). URL https://arxiv.org/abs/2001.08317.

[27] Rajamani, R. *Vehicle Dynamics and Control* (2006).

[28] Zhou, Y., Barnes, C., Lu, J., Yang, J. & Li, H. On the continuity of rotation representations in neural networks (2018). URL https://arxiv.org/abs/1812.07035.