IN-CONTEXT LEARNING OF INTUITIVE PHYSICS IN TRANSFORMERS

Cristian Alamos
University of California, Berkeley
Berkeley, CA
{calamos@berkeley.edu}

Masha Bondarenko
University of California, Berkeley
Berkeley, CA
{mbondarenko@berkeley.edu}

Martin Guo
University of California, Berkeley
Berkeley, CA
{guozy@berkeley.edu}

Alex Nguyen
University of California, Berkeley
Berkeley, CA
{aohk0512@gmail.com}

Max Vogel
University of California, Berkeley
Berkeley, CA
{max-v@berkeley.edu}

ABSTRACT

Transformers can perform new tasks when prompted with unseen test examples without updating their parameters, a phenomenon known as in-context learning (ICL). We build upon the ICL literature by investigating whether transformers can learn intuitive kinematics in context and predict the future state of a physical system given a sequence of observations. In particular, we train a transformer to predict the next position of a sequence of coordinates representing bouncing balls and vary parameters such as the strength of gravity and elasticity of the balls. We then evaluate the model's performance on in-distribution and out-of-distribution parameter combinations and use RNNs and LSTMs as baseline comparisons. We find that transformers show significant ICL capabilities on in-distribution examples, surpassing the baseline models. Transformer models are also relatively robust to distributional shifts such as being exposed to unseen speeds in test time, but their performance degrades rapidly when Gaussian noise is injected into inputs.

***K*eywords** Transformer · In-Context Learning

1 Introduction

Transformers have demonstrated a remarkable capacity for adapting to novel tasks without the need for parameter updates—a phenomenon termed in-context learning (ICL). Notably, within simple function classes such as linear regression, transformers have exhibited near Bayes-optimal in-context learning capabilities. We extend the understanding of ICL by exploring the potential of transformers to learn intuitive physics in context. In particular, we train a decoder-only transformer to predict the location of three balls bouncing in 2D space given all previous locations of the balls. Concretely, our inputs are sequences of coordinates that represent vectorized positions of the balls, as well as their radius:

$$(x_1, y_1, r_1, x_2, y_2, r_2, x_3, y_3, r_3)_1, \cdots, (x_1, y_1, r_1, x_2, y_2, r_2, x_3, y_3, r_3)_{L-1}$$

Our output sequences contain the predicted positions of the balls:

$$(x_1, y_1, x_2, y_2, x_3, y_3)_2, \cdots, (x_1, y_1, x_2, y_2, x_3, y_3)_L.$$

The prediction for the n-th frame is made using all $n-1$ previous frames. Our loss function is simply the average Euclidean distance between predicted and true positions. We vary parameters such as gravity strength and ball elasticity in training. We then evaluate the model's performance on novel combinations of these parameters.

Building off of Garg et al., 2020, we propose the following mathematical framework for thinking about ICL in the context of discrete-time dynamical systems depicting a bouncing ball. Let x_t be the state (position and velocity of the ball) and be a two-dimensional vector representing the parameters of the state-transition function (gravity and

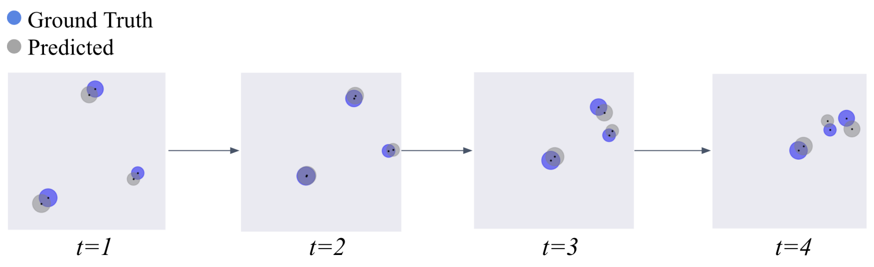


Figure 1: Visualization of input and output sequences. The inputs and outputs of the model are not themselves images, but rather sequences of coordinates that can have been visualized above for illustrative purposes.

elasticity). These stay constant within sequences but can vary across sequences. A prompt $\mathcal{P}$ is simply a recurrence relation $x_1, x_2, ..., x_L$, where $x_{t+1} = f_\theta(x_t)$.

Let D_X be a distribution over the initial states of the system. In our case, this is a distribution over the velocity and initial position of the ball. Let D_F be a distribution over the class of state-transition functions $\mathcal{F}$. In our case, this is a distribution over the strength of gravity and the elasticity of the ball.

We say that a model $\mathcal{M}$ can in-context learn the function class $\mathcal{F}$ up to ϵ, with respect to (D_F, D_X), if it can predict $f(x_L)$ with an average error $E_P[l(M(P)), f_\theta(x_L))] < \epsilon$.

2 Previous Work

Our project is rooted in the principles of in-context learning, as outlined in Garg et al.'s seminal paper, "*What Can Transformers Learn In-Context? A Case Study of Simple Function Classes*" [1]. The study delved into transformers' capacity to grasp and utilize contextual information, focusing on simple function classes. Through a case study, the authors explored how transformers enhance learning outcomes by leveraging contextual dependencies in input data. This research sheds light on transformers' strengths and limitations in capturing in-context information, contributing to a broader understanding of their applicability and effectiveness, especially in scenarios involving simple function classes.

Expanding into more complex analyses with In-Context Learning, "*Transformers Predicting the Future: Applying Attention in Next-Frame and Time Series Forecasting*" by Cholakov & Kolev [2] laid the groundwork for integrating transformer architectures into video prediction and time series forecasting. This paper demonstrated the effectiveness of transformers in capturing long-range dependencies within temporal data, specifically in predicting the next frame in video sequences. By addressing both next-frame prediction and broader time series forecasting, the research provided valuable insights into the adaptability of transformer models for diverse sequential tasks, forming a solid theoretical foundation for our investigation.

In the realm of realistic video prediction, Shouno's work in "*Photo-Realistic Video Prediction on Natural Videos of Largely Changing Frames*" [3] tackled challenges related to generating accurate predictions in dynamic video scenarios. The paper proposed techniques for handling largely changing frames in natural videos, enhancing photo-realistic video prediction. This research informs our approach to context learning in the presence of dynamic visual elements, offering insights into strategies for improving the realism and accuracy of predictions in complex relational image sequences.

3 Methods

3.1 Dataset Generation

To precisely control which combinations of parameters the transformer is trained on, we synthetically generate our data using the Pymunk physics library [4] in Python.

We generate two datasets, each consisting of 100,000 sequences of length 50.

The first dataset depicts three balls of varying radii in a 100x100 grid moving at a constant velocity. There is no gravity and all balls have a coefficient of restitution of one, meaning that they lose no energy after collisions. Each ball spawns in a random location moving in a random direction. This dataset is created to test how each model will generalize to out-of-distribution (OOD) examples when elements such as varying speed and gravity are introduced.

The second dataset is intended to represent a more complex and realistic physical environment. As in the first dataset, there are three balls in a 100x100 grid that spawn randomly in a random location moving in a random direction. However, the balls randomly vary in their initial speed, mass, and coefficient of restitution. In addition, the strength of gravity can vary across sequences. There are also three randomly-positioned static triangles in each grid which the balls bounce off of.

To incorporate all of this information, each frame is represented by a 27-dimensional vector. The first nine dimensions represent the coordinates and radii of each ball, while the last 18 describe the positions of the vertices of all three triangles. When inputting sequences from the first dataset into the model, the last 18 elements of each vector are set to 0. Information such as the mass and coefficient of restitution are intentionally excluded from the input as these are not identifiable by simply looking at an object. The model should infer these parameters by looking at the collisions.

All randomly sampled elements are drawn from uniform distributions, allowing for a more concrete understanding of what constitutes an OOD example. To illustrate this point, consider that it is not clear whether a speed of 3 is OOD for a dataset trained on a normal distribution of speeds with a mean of 0 and standard deviation of 1. However, if training speeds are sampled from a uniform distribution over [-2, 2], it is unambiguously OOD.

3.2 Model Architechture

We use a decoder-only architecture based on GPT2 provided by the HuggingFace library [5]. Following Garg et al. (2023), we compare our model to the two most well-known models in sequence-sequence tasks, namely LSTMs and RNNs. In addition, we compare our model to two manually implemented models. The first one simply predicts that each ball will be exactly where it was in the previous frame, while the second uses the last two frames to impute the speed and direction of the ball and assumes it keeps moving in that direction at that speed.

For the transformer, we use an embedding layer with an output dimension of n_{embed}, 2 decoder layers, 1 attention head, and a linear layer at the end. We set n_{embed} to 128 for training on the first dataset and 32 for the second dataset. Similarly, the LSTM and RNN have a linear layer for their outputs. These two models have 2 and 4 layers, respectively, and have $n_{embed} = 32$.

4 Results

4.1 Performance on In-Distribution Parameters

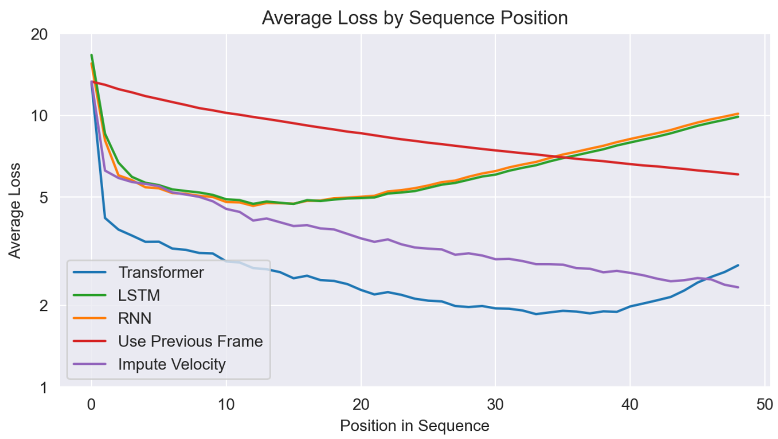


Figure 2: Performance by position in sequence across all models. The y-axis represents Euclidean loss averaged across 10,000 test examples on the second dataset.

As shown in Figure 2, all trained models show a sharp decline in loss in the first few sequences, indicating that they are learning the trajectory of the balls in context. This is to be expected, as it is possible to know the initial speed and direction of the balls only after the second frame and the acceleration only after the third. Similarly, the restitution and mass of the balls can only be inferred after their first collision.

Interestingly, the loss of the RNN and LSTM models begins to increase after roughly the 15th element in the sequence, while the transformer's loss increases after approximately the 40th example. One potential explanation for this phenomenon is that as sequences progress, the entropy of the system increases and the balls move slower and have more collisions with the ground, making the prediction of subsequent positions more difficult.

It is also clear that the transformer model outperforms all other models. In the very last frames, however, the model based on inferring velocity from the previous two frames has a slightly lower loss than the transformer.

4.2 Performance on Out-of-Distribution Parameters

Figure 3 demonstrates that transformers are less robust to the injection of Gaussian noise into test examples than to baseline models. While loss remains relatively stable for baseline models, the performance of transformers degrades linearly.

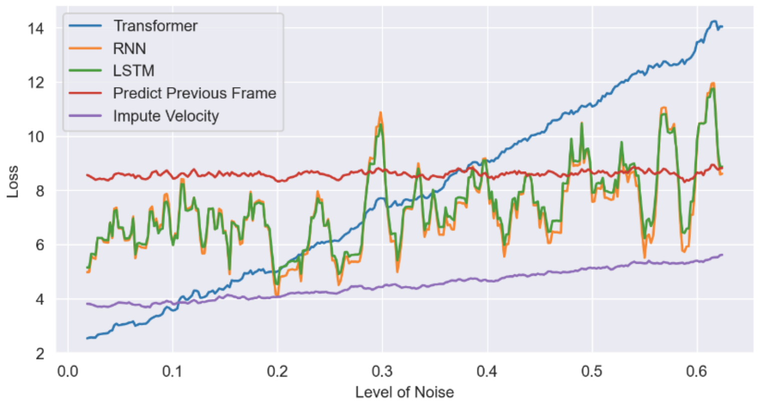
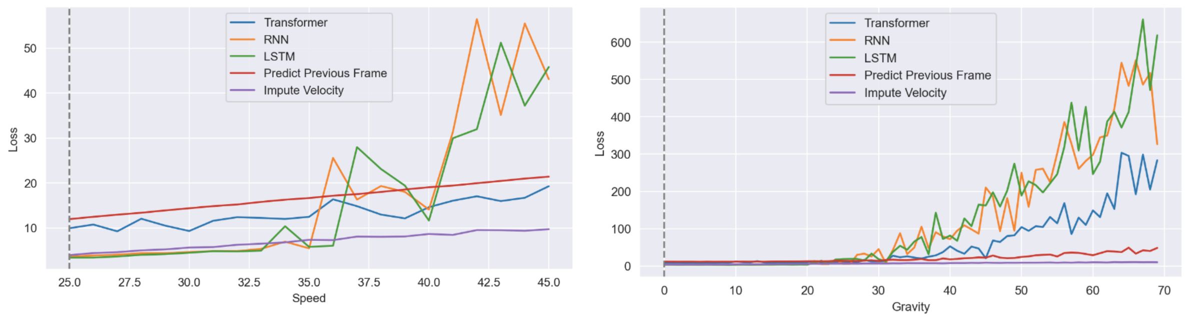


Figure 3: Model robustness to noise.



(a) Models trained with speed = 25

(b) Models trained with no gravity.

Figure 4: Robustness to distributional shifts.

Figure 4 shows that transformer models are more robust than RNNs and LSTMs to shifts in the speed of balls and the strength of gravity. In particular, 4a demonstrates that transformers trained on a dataset where balls move at a constant speed of 25 will generalize well to being exposed to high speeds at test time. In contrast, the loss of RNNs and LSTMs increases rapidly when the speed is increased above 35. 4b illustrates a similar pattern for changes in gravity, although the difference between transformers and the trained baseline models is less stark.

5 Future Work

Future work could evaluate vision transformers' in-context learning capabilities by using sequences of images rather than sequences of coordinates.[1]

In addition, a systematic exploration of which out-of-distribution examples lead to increased and decreased relative performance could help ground our findings within a theoretical framework.

[1]An earlier version of this paper attempted to use vision transformers, but this proved too computationally expensive.

Lastly, future work could investigate transformers' ability to simulate physical systems using autoregression. Specifically it would be valuable to investigate at what time point the model's autoregressive estimates diverge significantly from the ground truth.

6 Conclusion

We have shown that transformers display significant ICL capabilities on in-distribution examples, surpassing the baseline models like RNNs and LSTMs. Their performance is relatively robust to distributional shifts like being exposed to unseen speeds in test-time, but performance degrades rapidly when Gaussian noise is injected into inputs.

Thus, transformers' capacity to in-context learn intuitive physics in simple settings suggests their ICL capabilities could extend to more complex real-life environments.

References

[1] Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes, 2023.

[2] Radostin Cholakov and Todor Kolev. Transformers predicting the future. applying attention in next-frame and time series forecasting, 2021.

[3] Osamu Shouno. Photo-realistic video prediction on natural videos of largely changing frames, 2020.

[4] Victor Blomqvist. PyMunk, November 2023.

[5] HF Canonical Model Maintainers. gpt2 (revision 909a290), 2022.

A Responses to Reviewers

Thank you to all reviewers for providing very helpful feedback. We've addressed each of the main comments from the reviews below. Please note too, we opted to not color-code our modifications in the main paper as we significantly redid portions of our paper based on relevant feedback and challenges we faced, and it would not have proved helpful to make an attempt to nitpick what was not newly added.

Specifically, we made architectural and hyperparameter changes to the models that allowed the transformer, LSTM, and RNN to achieve much lower loss. We were therefore able to modify the training data to reflect a more complex physical environment depicting the movement of multiple balls. In addition, the models achieved satisfactory performance on the most complex training set without the need for curriculum learning, so we have omitted this section from the final draft.

"One caveat they brought attention to was the fact that increasing the length of the prompt did not increase the model's performance. This is contrary to the standard notion that larger prompts should improve the ability of the transformer to learn the environment." We agree this was a puzzling finding. In the most recent iteration of our paper, this phenomenon is greatly mitigated, though still present in LSTMs and RNNs.

"There are too few model comparisons to baseline to compare performance on a task with more complex models." We, too, recognize the problem of having few strong enough baselines to reliably compare performance of our model. We are mainly constrained by compute time and compute power, but we believe with these limitations, comparing our transformer model performance to an LSTM and RNN, as in the same vein of the original ICL paper [1], is sufficient.

"I agree with the author's comment on operating on images rather than coordinates, as this would allow for the model to learn more general dynamics involving other factors. I also agree with the author's claim that they should investigate the autoregressive pathways to understand how much input the model needs, since the paper isn't able to answer why the strength of the transformer doesn't increase with sequence length." In regards to this, we have further analyzed autoregressive pathways, but so far, we have not been able to achieve better performance.

"In terms of reproducibility, the paper attaches the code and data and states that the results are reproducible using Google Colab with a GPU, so everything needed to reproduce the results are given. The methodology is clearly detailed and all resources needed are provided. It's implied that rerunning the experiments and reproducing the results in the paper should be possible given the available code and methodology. However, when attempting to reproduce the results, I found it slightly challenging to actually reproduce them, even though everything was right in front of me. For example, I had to copy all files into my own google drive because I was unable to run some of the code provided because I didn't have the right permissions. A document with instructions about how to reproduce the results would be very helpful, and a GitHub repo would also be helpful." Our concerns with hosting a Github repo would be that people would have more

problems cloning our large files locally and running them locally as well. We utilized Google Colab for the GPU's as well, so our results would have been only reproducible there. We understand the problems in running the individual notebook file, and as such will create editable duplicates of our relevant notebook files so that people will not have to fully download our code or accidentally modify our code, and will also have outputs from our notebook to reference.

"To improve the paper, the authors could consider optimizing the transformer model performance further, possibly through hyperparameter tuning, to figure out why the model plateaus based on sequence length and if there is a way to avoid it. Another possible improvement is the exploration of more complex datasets with additional variables such as obstacles or multiple balls. Finally, the addition of a readme file in the code folder would make the reproduction process simpler for readers." We have added a README file to instruct individuals to look at the train.ipynb file for better guidance. We have looked into hyperparameter tuning, and we will elaborate more on our results in the final report.

"One weakness was in describing the models. You mentioned that the transformer is based on the "Attention is All You Need" paper, but I was not sure of the exact model parameters. The RNN and LSTM models were also not fully explained, so having an estimate of the parameters of all of these models would have been helpful in comparing them in terms of computing power." Now addressed in the final report in Section 3.2!

"Another weakness was in the dataset generation section. The paper mentions using 16x16 images for the transformer to learn. However, the transformer seems to be actually using a 28x1 location tuple. Later on, this is shown to be a future idea in the future works section, but within the dataset generation section, it seems confusing. This is compounded by the fact that the introduction presents a different input which are just (x,y) pairs." Resolved. More information in 1

"One suggestion I have is to possibly combine the curriculum levels while training the transformer in one go. The in-context learning paper mentions using curriculum learning to train a single model, as it would increase the speed and robustness of the transformer. This may be hard because you have discrete levels of training, but it may help improve the performance. " Resolved. We found that curriculum learning did not improve performance on the model, so we trained the model on the full range of complex data.

"The paper is somewhat reproducible. The code seems logical, however, the provided notebook was not able to be run without changes. I had to download the code and upload it to the notebook in order for it to register the required libraries. The output of the provided toy model seems similar to the paper's general thesis." Resolved.

"I think a different curriculum system could possibly improve the performance of your model. I am not sure how you incorporated variable parameters like variable gravity in your dataset, but it seems likely that these changes could have accounted for the large swings in the plots of your results section." We found issues in our code that had accounted for this large swing in our results, and now have plots that account for a gradual increase in performance across curricula.

"It would definitely be interesting to incorporate CNNs and processing images, however it does seem counteractive to the idea of a toy example." We originally investigated the work of CNNs pre-processing to the images before the encoder, but had architectural challenges that proved it wasn't effective.

"Also, investigating the boundary lines of how your model performs would be a very interesting followup to what you currently have." Addressed in the final report.

"Get data which has longer sequence to make the model be more robust." Resolved robustness in alternate ways in the final paper. We found otherwise that utilizing longer sequence lengths did not improve performance.

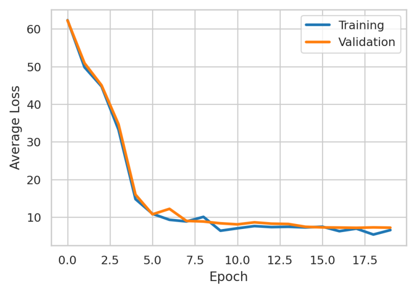
"The weaknesses include limitations in the complexity of experiments due to constraints in computational power and the need for additional optimization of the transformer model." Limitations to computational power and time, while still constraints, were worked around to generate new results.

"One suggestion for improvement is to provide more detailed explanations of the experiments and their results in the paper text, rather than relying solely on code and visualizations." Now addressed in the final report throughout and more so in 3.2.

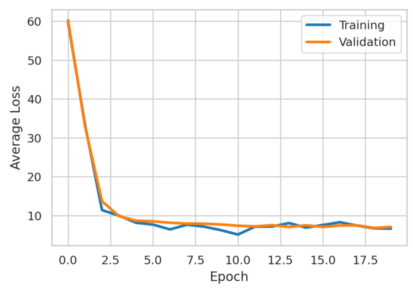
B Supplemental Figures and Analyses

Figure 5 shows overall performance on training and validation test sets. Note the scale of the axes. The Transformer loss was drastically lower than that of the RNN or LSTM in the first few epochs despite fluctuating downwards unevenly.

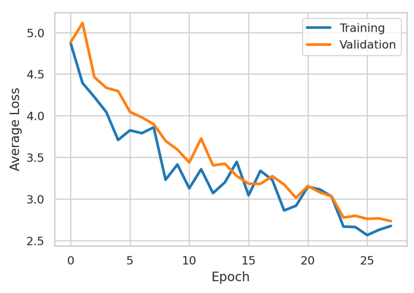
Figure 6 go on to show attention maps for heads in different layers of the transformer trained on data with constant speed. Figure 6a shows that the attention mechanism looks towards past sequence inputs more so than the second layer in 6b. In the latter, the attention mechanism focuses more closely on the directly previous sequence. This can be



(a) RNN



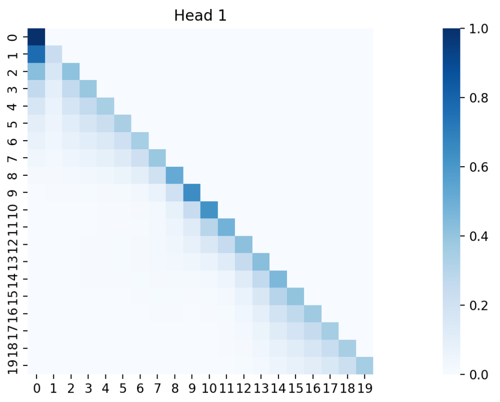
(b) LSTM



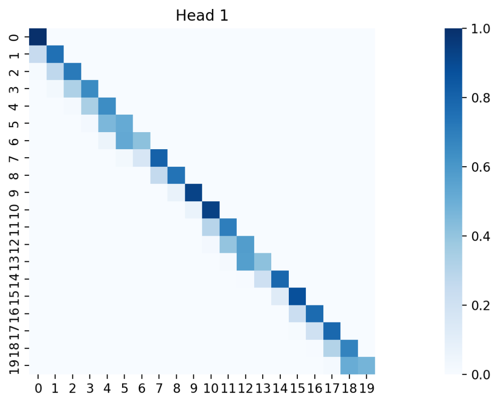
(c) Transformer

Figure 5: Full Trained Model Performance.

interpreted as the first layer looks more towards the embeddings of the input vector, or the positions, whilst the second layer looks towards higher level features between 1-2 timepoints.



(a) Attention Map for First Layer



(b) Attention Map for Second Layer

Figure 6: Attention Map for Transformer Trained on Dataset with Constant Speed