See Clearly, Act Intelligently: Transformers in Transparent Environments

Omar Elamin
Supervisor(s): Frank A. Oliehoek, Jinke He
EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Omar Elamin
Final project course: CSE3000 Research Project
Thesis committee: Frank A. Oliehoek, Jinke He, Mathijs de Weerdt

***Abstract*—Traditionally, Recurrent Neural Networks (RNNs) are used to predict the sequential dynamics of the environment. With the advancement and breakthroughs of Transformer models, there has been demonstrated improvement in the performance & sample efficiency of Transformers as world models. The focus has been on partially-observable environments where their capabilities can be maximally utilised. In this paper, we sought to investigate the conditions under which transformers outperform RNNs given a fully observable environment where states obey the Markov property. This provides insight into transformers' generalisation and predictive capabilities. Specifically, our experiments explored the impact of model complexity and the size of the dataset. We observed that transformers did not outperform our baseline implementation when given up to 7000 episodes of trajectory data. It was also observed that having shorter sequence lengths had a negligible impact on the performance of the model, leading to our recommendation of avoiding using transformers in these fully observable environments.**

I. Introduction

Transformers are the state-of-the-art (SOTA) technology responsible for the recent breakthroughs in Natural Language Processing (NLP) such as the Generative Pre-trained Transformer (GPT) [1] & Bidirectional Encoder Representations from Transformer (BERT) [2]. Transformers excel in handling sequential data due to their ability to capture long-range dependencies through self-attention [3]. An agent's actions in reinforcement learning are sequential by nature, and the current state of the art is to use Recurrent Neural Networks (RNNs) to capture this [4]. RNNs suffer from exploding & disappearing gradients, which is exacerbated when trying to capture longer-range dependencies [5]. Transformers do not have this limitation.

Furthermore, Transformers as applied to NLP are trained on words. Language is rich in features and intricate patterns, and the models are trained using a self-supervised learning objective. These properties are directly transferable to learning world models in Model-Based Reinforcement Learning (MBRL), making transformers a potential replacement for traditional RNN implementations.

Several studies have explored this applicability. For instance, Micheli, Alonso, and Fleuret [6] observed a 70% improvement in mean human normalised score on Atari 100k over Self-Predictive Representations (SPR) [7], the second best performing model, establishing a new state-of-the art for methods without lookahead. TransDreamer from Chen, Wu, Yoon, *et al.* [8] uses a Transformer State-Space Model in place of the Recurrent State-Space Model in DreamerV2 and matches its performance on most tasks, as well as outperforming it on complex tasks that require long-term, complex memory interaction. These studies assume a partially observable environment where the capabilities are largely utilised.

In fully observable environments where states satisfy the Markov property, the dynamics of the environment are known and sequences are shorter. This limits the utility of transformers and thus has not been adequately explored, highlighting a knowledge gap regarding the performance and utility of transformers in these environments. Specifically, it allows us to examine the capability of transformers in capturing past signals that give more informed insights for making predictions, and whether these signals can amount to giving enough information to allow the transformer to outperform a more traditional approach. Crucially, the benefit of identifying this is it allows us to understand if the added computational overhead is justifiable.

Using a baseline model based on DreamerV2 [4] & EfficientZero [9], We will investigate the question "Under what conditions do transformers enhance the planning performance of model-based reinforcement learning in fully observable environments compared to conventional methods?"

We will begin by introducing the background & the existing work related to our investigation (section II). We will then outline the details of the implementation of the transformer based on IRIS [6], as well as the evaluation and data collection techniques (section III). Then we will investigate how model size (subsection IV-E) & dataset size (subsection IV-F) impact performance.

II. Background

MBRL, as opposed to model-free reinforcement learning, involves building a model of the world the agent is operating in, to make more informed & ideally optimal decisions. Specifically, instead of directly learning a reward function $Q(s, a)$ for a state-action pair, we build a model that learns a deterministic *state transition* function $T(s, a) \rightarrow s'$ that outputs a resultant state s', and a reward function $R(S, a) \rightarrow r$ that outputs a reward. The following sections introduce some concepts relevant to the implementation used in the experiments.

A. Autoencoders

To process the states in a deep-network architecture, we must represent the information meaningfully to the model. The information related to the current state of the world is given as images, represented in RGB pixel data format. This format is largely non-descriptive for the world model's goals. Thus, a method of reducing the dimensionality of the input into a more descriptive latent space is necessary, and autoencoders are the SOTA [10].

An autoencoder is a symmetrical network architecture where the same input is used as the expected output to learn meaningful latent data representations. This can be seen in Figure 1, where the useful representation is learned in the central bottleneck portion.

A *discrete* autoencoder works similarly, but instead of learning the model in a continuous space, it learns a set of N discrete latents [12] and the output of the encoder part in Figure 1 is mapped to the closest discrete latent.

B. Recurrent State-Space Model

In DreamerV2, a "Recurrent State-Space Model" is used, combining the *stochastic transitions* of State-Space Models and the *deterministic transitions* in RNNs to split their problem

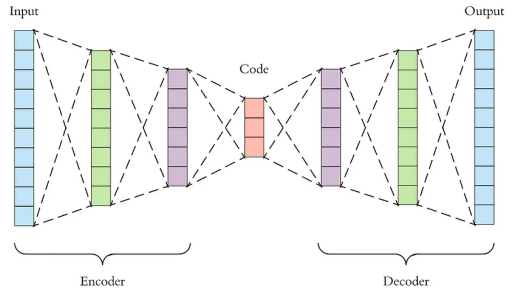

Figure 1: Architecture of an autoencoder, Adapted from: [11]

into deterministic & stochastic components. This recurrent state space model operates on the latent space and predicts dynamics from one latent representation to another [13].

Our environment is fully deterministic, and so the baseline model provided omitted this functionality, opting instead for what is functionally a recurrent dynamics model that learns the transitions $T(s, a)$ in a latent space.

C. Transformers

Transformers provide an alternative approach to modelling sequential dependencies in data, using a mechanism called *attention* popularised by Vaswani, Shazeer, Parmar, *et al.* [3] in 2017. Focusing on this form of *causal attention* in an *auto-regressive* transformer, it can be thought of as a method of looking back at past events and searching for different "patterns" of dependencies. Each search for a "pattern" is called an *attention head*, and collectively sending out h attention heads allows the model to gain h different types of insight into how the past affected the current state.

This insight is then incorporated into the latent representation of the current state, allowing the transformer to generate a distribution of possible next states using this newly enriched latent vector. Sampling from this distribution gives a possible prediction for the next state. This process is chained to generate prediction chains of arbitrary length. This chaining process is what makes the transformer "auto-regressive".

D. Transformers as World Models

The paper by Micheli, Alonso, and Fleuret [6] explores the applicability of transformers as world models. They use an auto-regressive transformer with a discrete autoencoder, where the dynamics of the world are modelled as sequences, captured by the transformer.

The state vectors input to the transformer are represented as a vector of indices $\mathbf{z_t} = (z_t^1, z_t^2, ..., z_t^K)$ corresponding to the discrete latents from an embedding lookup table $\mathcal{E} = \{e_i\}_{i=1}^N$. An input image x_t is passed through a CNN to produce an output y_t, which is mapped to the closest embedding s.t. $z_t^k = \mathbf{argmin}_i ||y_t^k - e_i||$. This discrete autoencoder method is taken from Oord, Vinyals, and Kavukcuoglu [12].

The transformer then takes an input sequence such as $(z_t^1, z_t^2, ..., z_t^K, a_t, z_{t+1}^1, ..., z_{t+1}^K, a_{t+1})$, which is a sequence of two state-action pairs, and processes it as described in subsection II-C. This is illustrated in Figure 2. In this figure, it can be seen that the world model predicts the reward r_{t+1} as expected, but it also predicts a terminal flag d_{t+1}, alongside the next state $\mathbf{z_{t+2}}$.

III. Methods

To learn a model of the environment and evaluate it in an offline setting, a few steps must be taken. Firstly, a dataset of an agent interacting with the environment is necessary for training. Secondly, a method of using the environment model to learn a policy function must be implemented for evaluation.

A. Deep Q-Network

Deep Q-Network (DQN) is a model-free reinforcement learning algorithm where the goal is to learn a policy that maximises the cumulative reward by approximating the optimal action-value function $Q(s, a)$ [14]. DQN primarily uses a deep neural network to estimate Q-values, combining the benefits of reinforcement learning and deep learning.

An agent selects actions using an ϵ-greedy strategy. This means that at each point the agent selects either a purely random action with probability ϵ, or the optimal Q-value with probability $1 - \epsilon$. This is annealed linearly to eventually lead the agent to choose optimal Q-values only.

This generates interaction data from the environment that can be used to train the world model. Using this approach provides high-quality & meaningful trajectories, rewards, and terminal data.

B. Monte Carlo Tree Search

It is also necessary to try to use the world model to make decisions in order to evaluate its competence. This can be done using Monte Carlo Tree Search (MCTS), which is a search algorithm used in decision-making processes, particularly effective in games and complex decision environments [15]. It works by building a search tree incrementally, using random roll-outs from its leaves to estimate the value, in turn helping determine the optimal decisions to take.

MCTS is used in the learned world model to search through the imagined search space and determine optimal actions. These actions can then be taken in the real environment and evaluated based on future rewards. This provides a numerical measure of the planning performance of the world model.

C. Transformer World Model

Most importantly, a choice of implementation for the transformer is necessary. This will be implemented using the transformer from IRIS, utilising the fact that it was developed for a very similar use case. The paper by Micheli, Alonso, and Fleuret [6] does not motivate using the discrete autoencoder and learning an index-based language in the transformer to represent the states. Therefore, in the interest of controlling the scope of the experiment, this step will be omitted.

We can therefore use the latent representation of the state $\mathbf{s_t}$ concatenated with the action yielding a vector $(s_t^1, s_t^2, ..., s_t^K, a_t)$. This vector can be passed through the embedding network similar to Figure 3 to yield a new

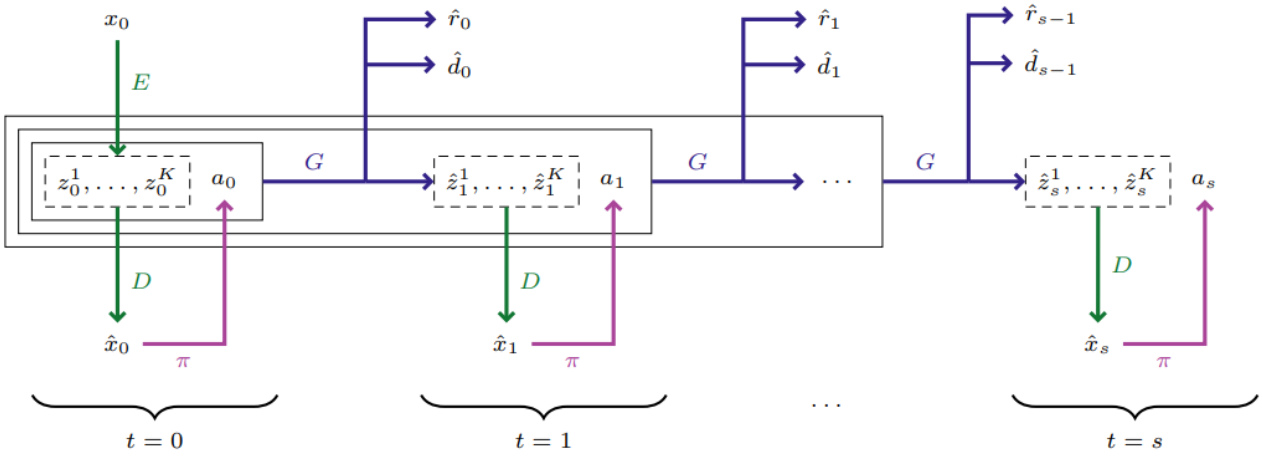

Figure 2: Transformer Architecture in IRIS, Adapted from: [6]. It can be see that the index set representing the state is paired with the action and then passed through the transformer producing a reward, done probability, and a next state. This can be paired with a new action to continue this *auto-regressive* prediction chain.

embedding $\mathbf{x_t}$. The transformer thus takes input sequences of the form $((x_t^1, ..., x_t^K), (x_{t+1}^1, ..., x_{t+1}^K), ...)$.

$$\textbf{PositionalEncoding} = \begin{cases} \sin(e^{d*i}) & \text{index is even} \\ \cos(e^{d*i}) & \text{index is odd} \end{cases} \quad (1)$$

This sequence is passed through a positional encoder. IRIS did not have a positional encoder as a part of their transformer implementation, so one was created for the purposes of this experiment. Positional encoders incorporate information about the state's position in the sequence into its vector, so that the attention heads can learn to look for them. The calculated positional encoding is simply added to the existing state vector and the result is used as input for the transformer. In Equation 1, d is a constant divisor term that depends on the embedding dimension, and i is the index. Essentially this encodes each position in the sequence in a unique way such that it is meaningful to the transformer. This approach is taken from Vaswani, Shazeer, Parmar, *et al.* [3].

$$\textbf{Loss} = w_{dyn} \cdot Loss_{dyn} + w_{rep} \cdot Loss_{rep} + w_{reward} \cdot Loss_{reward} + w_{done} \cdot Loss_{done} + w_{rec} \cdot Loss_{rec} \quad (2)$$

Training the transformer is quite involved, since sequences must be retrieved and we do not want to retrieve sequences that overflow into new episodes. The dataset was therefore grouped by episode, and each batch iteration a random sequence sample of length $sequence\_length$ is taken from the episode. Additionally, it was observed that the model was quickly getting stuck in local minima by minimising the dynamics loss and ignoring the others, since this could be achieved by simply making all embeddings close to zero. To solve this, we introduced weight parameters to the model for each loss term, and increased the weight of the reconstruction loss in order to learn the embedding first, then learn the dynamics based on this. The formula for the loss can be found in Equation 2, and these corresponding paramters in Table I.

IV. Experiments

Using the two implementations of world models, an investigation into what factors cause one model to outperform the other was employed. It was specifically desirable to investigate whether larger transformers are superior to a large baseline model and whether one improves at a quicker rate when more data is provided.

Performance is measured using the future rewards obtained by using the imagination of the world model in MCTS to take actions. This gives an indication of the planning performance of the world model. We are specifically interested in transformer's generalisation capabilities. Given limited data, the model ability to generalise to unseen states vs over fitting to its training set is an aspect that could provide insight into transformer's capabilities. The intuition is motivated by Transformer's demonstrated sample efficiency in partially observable envrionments [6].

A. Environment & Setup

The environment chosen for this research is the MinAtar breakout environment. This environment is minimal, with its images being only 10x10 in size, as well as being much simpler than the traditional breakout game. Given that this investigation uses transformers, which are heavily demanding models, it is infeasible within the scope of this research to use larger or more complex environments. Additionally, it is paramount to the core of the investigation that the environment is fully observable and Markov, which a simpler environment like MinAtar breakout satisfies.

B. Hypotheses

There are a couple factors to consider when hypothesising on how the models will preform. The first and major one is what the gained utility of looking at past states is. It is

reasonable to expect that past states might include some clearer signals that are useful for future state prediction, even if that signal is also present in the current state from the Markov property (though less clear). It is these signals that could give the transformer an advantage from looking into the past.

Additionally, the architecture of the transformer might provide some superiority over the baseline model. The attention heads may prove to be a more effective way of retrieving the relevant information from the past state(s) to predict the future state. This may be true even if we are only using timestep $t-1$ for future prediction.

However, it is also possible that these factors along with the fact that transformers are inherently larger models than a simple 3-layer MLP, that it is prone to over fitting to the data.

The hypothesis however for this experiment, is that the transformer will outperform the baseline when there is less data, due to its proven ability to be more sample efficient [6], but as more data is available, the baseline should match its performance.

In terms of model size, the transformer's architecture is inherently larger, and so is more capable of capturing more information and so the second hypothesis is that the transformer will outperform the baseline for larger latent space dimensions.

C. Baseline Model

To evaluate the transformer, there is a need for an implementation to be used as the baseline. The implementation provided for this purpose is a sequence model inspired by DreamerV2 [9] & EfficientZero [4]. This model has three components: a representation net E, a reconstruction net D, and a dynamics net M. E and D together make up the autoencoder, while M learns the transition function $T(s,a)$, the reward function $R(s,a)$, and a terminal function $d(s) \rightarrow [0,1]$ which determines a probability of whether taking an action at a given state yields a terminal. Specific hyperparameters can be found in Table II. Additionally, the code is available in the project repository.

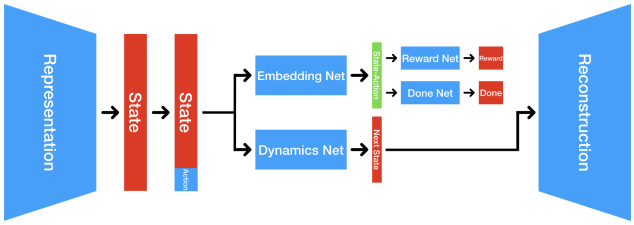

Figure 3: Architecture of the Baseline Model

D. Transformer Hyperparameters

Using the hyperparameter set from IRIS as a starting point, and tuning them to optimise performance & convergence of losses, the set of parameters in Table I were chosen. The decision was taken to fix the context sequence length parameter to 20 in order to meaningfully evaluate the use case of the transformer, despite the fact that low sequence lengths were displaying equal or better results (see subsection IV-G).

Hyperparameter	Value
Batch Size	128
Number of Epochs	500
Learning Rate	0.00025
Embedding Dimension	256
Sequence Length	20
Number of Layers	10
Number of Attention Heads	4
Dropout on Embeddings	0
Dropout on Attention	0
Dropout on Residual Connections	0
Weight Dynamics Loss	10
Weight Reconstruction loss	20
Weight Representation loss	0.1
Weight Reward loss	0.1
Weight Done loss	0.1

Table I: Hyperparameters used in Transformer implementation.

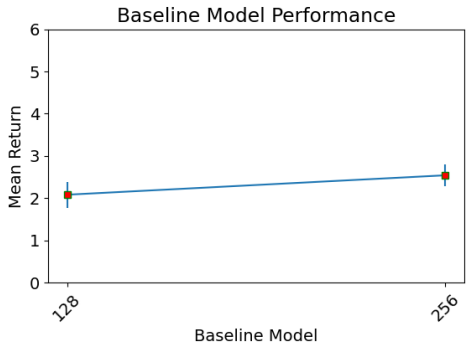

Figure 4: Mean Undiscounted Reward of Baseline Model with 7000 episodes. The ranges represent the 95% confidence interval.

E. Experiment 1: Model Size

Firstly, an investigation into how model size affected performance was conducted. Larger models can encapsulate more information, but can be susceptible to over-fitting and are harder to learn. A core part of the size of the model is the size of the embedding dimension, as it dictates the number of hidden features the models can use to represent the environment. This makes this variable the ideal candidate for an independent variable in a model size experiment.

It was observed that, firstly, the baseline model appears to perform better when the model size increased. This can be

observed in Figure 4, where the mean reward of an embeeding of 256 was 2.54, compared to 2.08 when the embedding was 128.

Conversely, the transformer seems to perform more poorly when the model size is increased. This can be observed in Figure 5, where an embedding size of 128 yielded a mean reward of 3.3, compared to 2.5 from 256.

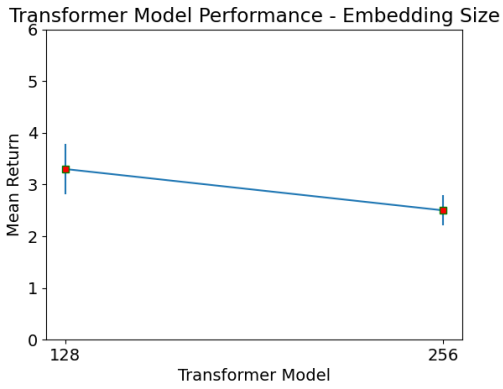

Figure 5: Mean Undiscounted Reward of Transformer Model with 7000 episodes. The ranges represent the 95% confidence interval.

F. Experiment 2: Dataset Size

The second experiment is related to changing the amount of data given a dataset of 7000 episodes. The models were trained on 1000, 3000, & 7000 episodes and evaluated for planning performance.

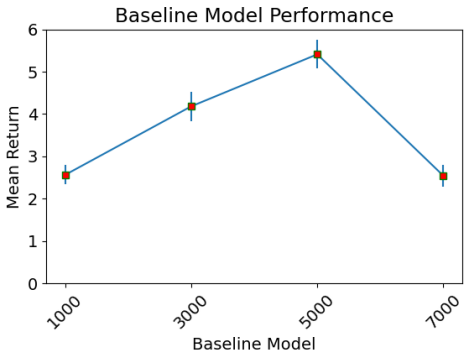

Figure 6: Mean Undiscounted Reward of Baseline Model with embedding size 256. The ranges represent the 95% confidence interval. Given the quicker evaluation runs of the baseline, it was also feasible to run it for the 5000 episode checkpoint.

As expected, more data proved to be more useful for both models. The transformer however seems to not be benefiting much after 3000 episodes (see Figure 7), with a slight but statistically insignificant decrease in performance from 3000 to 7000 episodes.

The baseline seems to perform worse after 5000 episodes, achieving 5.41 mean rewards at 5000 episodes, but with 7000 it achieves a mere 2.54. These results can be seen in Figure 6.

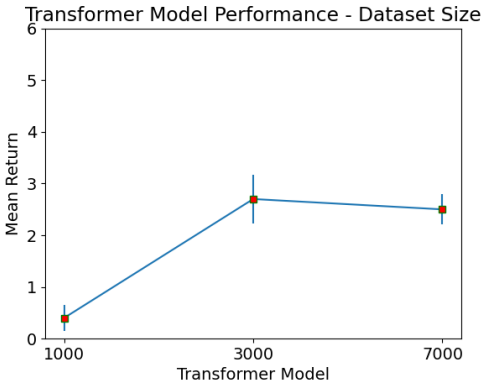

Figure 7: Mean Undiscounted Reward of Transformer Model with embedding size 256. The ranges represent the 95% confidence interval.

Hyperparameter	Value
Batch Size	128
Number of Epochs	30
Learning Rate	0.00025
Embedding Dimension	128
Dimension of Hidden Decoder Layers	64

Table II: Hyperparameters used in Baseline implementation.

G. Experiment 3: Sequence Length

In order to gain insight into the utility of transformer's sequence modelling strengths in these environments, investigating the impact of shorter sequences on the performance is useful. It was observed that shorter sequence had little to no impact on performance. Using sequence length 1 yielded the best performance seen from any of the models, 5.5 mean reward.

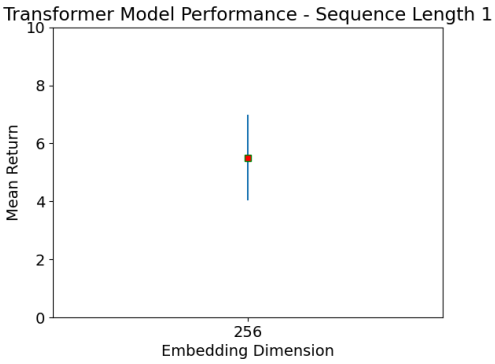

Figure 8: Mean Undiscounted Reward of Transformer using 7000 episodes. The ranges represent the 95% confidence interval.

V. Discussion

Having detailed the experiment and the resulting outcomes, this discussion section will provide a critical & insightful analysis of these results. For ease of comparison of the two

models, refer to Figure 9. For all its intents and purposes, usage of "statistically significant" refers to a 95% confidence interval.

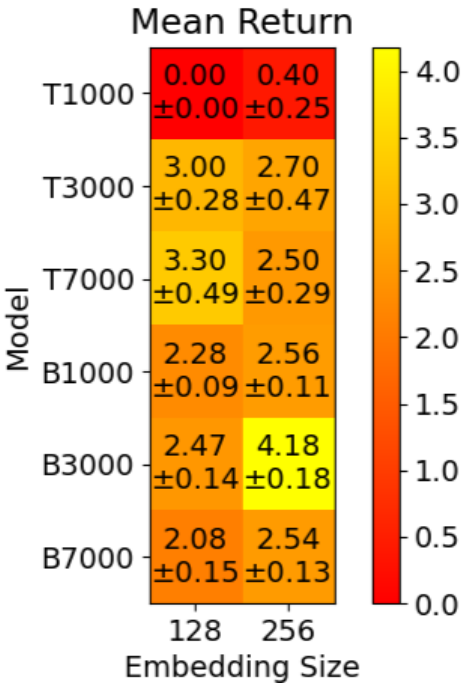

Figure 9: Heat map of Rewards and Std Errors for different episodes/embedding size combinations. T represents a transformer run, and B represents a baseline run.

A. Experiment 1: Model Size

The model size experiments showed minor changes in planning performance, with a statistically insignificant decrease in performance for the transformer, meaning we cannot conclude that it had any real impact on performance given our dataset. On the other hand, the baseline model did experience an increase in performance that was statistically significant.

This poses a challenge, since this same baseline model has shown superior performance with a 128 embedding dimension when more data is given to the model. One reason for this could be that the model is performing better with larger embedding sizes when there is less data since the model can try to over fit to what it has a little more effectively with a larger latent space.

B. Experiment 2: Dataset Size

The dataset size experiments analysed the impact of giving more data to the model on it's planning performance. It was observed that more data improved or did not change performance except in the case of of the baseline model when using an embedding of size 256, and going from 3000 episodes to 7000. The resulting rewards decrease by over 9 standard errors, which is clearly a significant loss. This is not the case for the same step in dataset size when the embedding size is 128, where the decrease in insignificant.

A possible explanation for this reduction is again referring back to this over fitting idea. Learning from more data leads to less over fitting to the original data, meaning the model is likely learning more information about states of the game that are later on, when behaviour is more optimal, but losing some accuracy in the early game in the process. This loss can be exacerbated when dealing with a larger latent space, and so could explain this significant reduction in accuracy.

C. Experiment 3: Sequence Length

The outcome of this experiment highlights a key potential issue with the approach of using the past in a fully observable environment; this procedure simply adds noise to the prediction. By limiting the sequence length to 1 and observing this remarkable performance it can be inferred that there is little to gain by looking in the past, and thus the variable that would have allowed transformers to excel is no longer a factor.

On the other hand, it also opens up the door to an exploration of different baseline models to capture the dynamics of the environment. Since it showcases that a different technique of using the last state to predict the future can outperform the baseline. This is not an indication of transformers being superior to traditional approaches, rather that there is room for improvement in the baseline model.

D. Limitations

These results highlight a significant limitation of the experiments, and that is amount of data. It is clear from these results that 7000 episodes is not enough to evaluate the efficacy of the model. The reason for the choice of 7000 is due to memory constraints in the hardware available, where 7000 episodes was using 16GB of RAM, and asking for more from the DHPC cluster would result in infeasible queue times. Future work should look at optimising the memory usage of the dataset loader for the transformer learning loop, and training the models with much more data. This was unfortunately not feasible within the time frame of this project, given the already implementation-heavy aspect of it.

Additionally, the results are sensitive to the random state of the program, due to various elements such as the random weight initialisation, dataset shuffling & splitting, and neuron dropout, among others. Ideally, one would run the experiments with several different seeds and ensure that the results are reproducible. Due to the time constraints of this project, and the lengthy training time of these models, this was infeasible.

Another limitation of these results pertains to the chosen baseline model. Although it was developed using established methodologies, there is an inherent limitation in that it may not accurately represent the state-of-the-art approach. It is possible that an alternative architecture for the baseline model could significantly outperform the transformer.

Due to the scope of this project, the MinAtar environment was chosen for its simplicity. This provides a valuable initial

probe into the capabilities of these models, however, there is an intrinsic limitation in this simplicity in that it raises concerns regarding the scalability of the results.

E. Future Work

Given these limitations, and to ensure that the results are robust and broadly applicable, future work should explore the areas of reproducibility, choice of baseline model, scalability, transformer enhancements, and real-world deployment.

In terms of reproducibility, it is recommended to run experiments with different random seeds to assess the stability of the results and mitigate the sensitivity to randomness.

Regarding the baseline model, future work could explore variations such as Convolutional Neural Networks or Long-Short Term Memory models. Additionally, more extensive hyper-parameter tuning could be carried out to find a more optimal set of parameters for the current baseline model.

Considering MinAtar's simplicity, future work should explore more complex environments, such as a non-simplified version of Atari Breakout, to reinforce the reliability of the results found in this study.

Furthermore, investigating different types of transformers, particularly those incorporating memory mechanisms or hierarchical attention, could potentially enhance performance further.

Finally, to understand the practical utility of these techniques, they should be deployed in real-world scenarios. This will provide a more concrete evaluation of their performance beyond theoretical justifications.

VI. Responsible Research

To ensure that this research is being conducted responsibly, it is important to analyse both the reproducibility & integrity of the contents of the paper. Additionally, it is imperative to consider the contexts in which the findings of this paper could be used in a way that is harmful to individuals or society at large.

For full reproducibility, all code used to generate any results in this paper is available in the project repository. Additionally, all hyperparameters used to train the models are included in the appendix, and the architecture is outlined in detail in the paper. The model weights are also available for download from the project repository.

All steps taken have been motivated using referenced literature, and the limitations of the results have been discussed in section V to ensure maximal integrity of the results arrived at in this paper.

Artificial Intelligence is a topic of much deliberation in moral philosophical circles and society at large. The nature of the contents of this paper being publicly accessible gives it the edge of democratising the technology. While it potentially allows for malicious actors to use it for undesirable purposes, ultimately it aids in the goal of ensuring that no one entity has a monopoly on its capabilities.

Additionally, the proposal that transformers are a superior alternative to RNNs encourages replacing the usage of RNNs with transformers. Transformers require much more energy and are therefore more detrimental to the environment. It is therefore our recommendation to prioritise maximising model efficiency. Additionally, we strongly urge using energy sources with the lowest carbon footprint available.

VII. Conclusion

The exploration of transformer models in fully observable, Markovian environments, where the state transitions are deterministic and predictably governed by current state actions, reveals critical insights into the applicability and performance of these advanced models. This thesis scrutinised the efficacy of transformers compared to a baseline recurrent model under varying conditions of dataset size and sequence lengths.

The experimental results indicate that transformers do not outperform the baseline model when data is limited. This showcases the challenges transformers face in environments where the ability to extract and make use of complex patterns or dependencies from data is limited. Furthermore, the investigation into the impact of sequence length has demonstrated minimal influence on the performance outcome. This finding suggests that in scenarios where the Markov property holds, the potential benefits of transformers, which are typically realised in handling longer and more complex sequences, are negated.

The implications of this study are twofold. First, it cautions against the indiscriminate application of transformers in environments where the prerequisites for their success—such as large datasets and the presence of complex, long-range dependencies—are not met. Second, it invites a reevaluation of the computational overhead and resource demands associated with transformers, especially in contexts where simpler models could achieve comparable or superior performance.

In conclusion, while transformers present a significant advancement in modelling sequential data in many complex scenarios, their application within fully observable, Markovian environments does not justify the additional complexity and resource requirements. Future research should continue to explore the boundaries of effectiveness for transformer models, potentially exploring hybrid approaches that could leverage their strengths while mitigating their limitations in data-constrained scenarios.

References

[1] T. B. Brown, B. Mann, N. Ryder, *et al.*, *Language models are few-shot learners*, 2020. arXiv: 2005.14165 [cs.CL].

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019. arXiv: 1810.04805 [cs.CL].

[3] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2017. arXiv: 1706.03762 [cs.CL].

[4] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba, *Mastering atari with discrete world models*, 2022. arXiv: 2010.02193 [cs.LG].

[5] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994. DOI: 10.1109/72.279181.

[6] V. Micheli, E. Alonso, and F. Fleuret, "Transformers are sample-efficient world models," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=vhFu1Acb0xb.

[7] M. Schwarzer, A. Anand, R. Goel, R. D. Hjelm, A. Courville, and P. Bachman, *Data-efficient reinforcement learning with self-predictive representations*, 2021. arXiv: 2007.05929 [cs.LG].

[8] C. Chen, Y.-F. Wu, J. Yoon, and S. Ahn, *Transdreamer: Reinforcement learning with transformer world models*, 2022. arXiv: 2202.09481 [cs.LG].

[9] W. Ye, S. Liu, T. Kurutach, P. Abbeel, and Y. Gao, *Mastering atari games with limited data*, 2021. arXiv: 2111.00210 [cs.LG].

[10] Q. Fournier and D. Aloise, "Empirical comparison between autoencoders and traditional dimensionality reduction methods," Jun. 2019, pp. 211–214. DOI: 10.1109/AIKE.2019.00044.

[11] A. Dertat. "Applied Deep Learning - Part 3: Autoencoders." (Oct. 2017), [Online]. Available: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798.

[12] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, *Neural discrete representation learning*, 2018. arXiv: 1711.00937 [cs.LG].

[13] D. Hafner, T. Lillicrap, I. Fischer, *et al.*, *Learning latent dynamics for planning from pixels*, 2019. arXiv: 1811.04551 [cs.LG].

[14] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, *Playing atari with deep reinforcement learning*, 2013. arXiv: 1312.5602 [cs.LG].

[15] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," vol. 4630, May 2006, ISBN: 978-3-540-75537-1. DOI: 10.1007/978-3-540-75538-8_7.