Analysis of spacial and temporal awareness of the World Models Agent

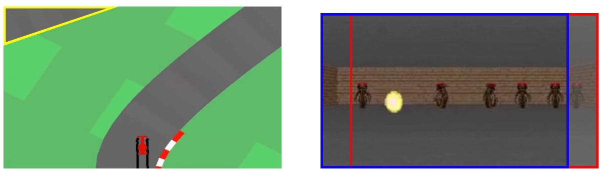
Rafael Bischof, Constantin Le Cleï, Dušan Svilarković, Steven Battilana

January 18, 2020

Abstract

We explore a variation of the World Models architecture to visualize if intuitive human spatial and temporal representations are indeed captured by the reinforcement learning agent. We propose to enhance the VAE training such that it learns how to enlarge, in essence zooming out, given input frames. Using a scaling factor on input frames the VAE managed to learn how to scale an image and even predict a curve that was outside of the original image. Additionally, we managed to isolate the latent feature that provoked the scaling. We also verified that the RNN leaned the temporal information, and is for instance able to remember a fire ball that left the frame but reappears when enlarging the image, which the VAE failed to do.

Introduction



(a) Car racing game [Bro+16] with highlighted part of the road announcing a narrow curve

(b) Two succeeding frames showing an Agent's movement to the left in Doom [Hak19]. Red frame at time t, blue frame at time $t + 1$

Figure 1: Car racing and Doom (Zombie) game.

World Models [HS18] is a Deep Learning (DL) method, inspired by our own human cognitive system [RS19], in order to do model-based reinforcement learning [PN17] (RL). The model (Agent), proposed by the authors David Ha and Jürgen Schmidhuber, consists of a variational autoencoder (VAE) [KW13], a recurrent neural network [HS97] (RNN) and a single-layer linear network (Controller).
The Agent's components all have a specific task:

- The VAE compresses the visual information into lower dimensions to make it more interpretable to the following components.
- The RNN receives the compressed frames as input and is trained to predict the next frame.
- The Controller observes the RNN's latent state and decides on an action, e.g. press a button.

Each component is being trained separately: the VAE and RNN using stochastic gradient descent and the controller using covariance matrix adaptation evolution strategy (CMAES) [HS18].
As its name suggests, a World Models Agent is able to understand the properties of an environment that are necessary to fulfill a certain task. It learns the physics and probability distributions and can therefore model the near future by looking at past observations.
In order to achieve good performance, many RL tasks require the Agent to very accurately model its surroundings, even beyond the frame's border. In the situation shown in Figure (1a) of the car racing game [Bro+16], the agent must anticipate a narrow curve when seeing the part of the road highlighted in yellow. Another example is when moving left in the Doom Zombie game [Hak19] in Figure (1b), it needs to remember monsters and fireballs that temporarily leave the screen in order to not get hit once it moves back right. But how good is an Agent's model about its surroundings really?
To investigate this question, we extended the World Models'[1][2][3] architecture with various methods as will be explained in more detail in Chapter Methods.

Related work

Learning a model of the world has been of great interest to help relieve the controller from having to directly map visual/game observation to action. Having it instead predict based on a processed latent representation of the game state helps speed up its learning. The World models paper [HS18] achieves this goal by passing the controller an encoded frame that contains visual and temporal information, and constraining the controller to a very simple neural network architecture. Using a recurrent part to model the environment, as

[1]Code provided by D. Foster (Applied Data Science) at `https://github.com/AppliedDataSciencePartners/WorldModels` [Dav19]

[2]Code provided by Hardmaru at `https://github.com/hardmaru/WorldModelsExperiments/tree/master/doomrnn` [har18]

[3]Code provided by S. Hakenes at `https://github.com/shakenes/vizdoomgym` [Hak19]

is done in this model, was first introduced in Making the World Differentiable [Sch90b], and further researched in [Sch90a] and [MW91]. In contrast to training each component, i.e. VAE, RNN, and controller, Risi and Stanley trained World Models end-to-end using genetic algorithm achieving comparable results as in the original World Models paper [RS19]. Freeman et al. demonstrated that it is possible to train the agent to perform well in its environment without forward prediction. They only pass the real observation with probability p, otherwise input the models own past internal generated observation [FMH19].

Methods

In [HS18], the World Model agent is trained in stages by successively pretraining the VAE on random rollouts, then the MDN-RNN in the same manner, and finally the controller. In the end, the controller takes as input the current encoded frame and reward to produce an action. In our model, we are mostly interested in visualizing the frames produced by the VAE and RNN, not on the score of the agent. We have therefore ignored the controller part since random rollouts are sufficient.
For the sake of spatial extension, we give the Agent a frame and get back a larger (zoomed out) image, with the additional area filled with what the Agent expects to be there. To achieve this we propose to modify the training set by cropping the frames by a fixed amount α and letting the Agent predict the original frame. More specifically, we take the game's frames of size (n, m) and convert them into pairs of images, where the first image (input) is the frame cropped to the size (n', m') and then scaled back up to (n, m), where $n' = \alpha n$ and $m' = \alpha m$, $0 < \alpha \leq 1$, and the second image (output) is the original frame (n, m).
A model trained on this modified dataset is able to take an image of size (n', m') and predict an image of size (n, m). But our ultimate goal is to pass it the original frame (n, m) and receive back a larger image $\left(\frac{n}{\alpha}, \frac{m}{\alpha}\right)$. The reason why this is not going to work is that the model never learned to enlarge anything else than (n', m') images. In the car racing game it will learn that the road in the input images is for instance 30px large and that it has to be 15px large in the target images. But it will be confused when receiving an image with a road that is already 15px large.
For our model to learn the notion of scaling, we also introduce a β_i, $0 < \beta_i \leq 1$ for each sample i, crop both images in the pair such that their sizes become $(\beta_i n', \beta_i m')$ and $(\beta_i n, \beta_i m)$ respectively and finally scale them back up to their original size (n, m). Note how the difference of zoom between the small images $(\beta_i \alpha n, \beta_i \alpha m)$ and the large images $(\beta_i n, \beta_i m)$ stays always α, even when introducing β.
This variation of zoom levels in the dataset ensures that the model learns what it means to enlarge images by a factor α^{-1} instead of learning fixed sizes and shapes. This will ultimately allow us to enlarge the original frames, although, the model has only been trained on smaller images.
We found a good choice of β to be $\beta = \alpha + \gamma(1 - \alpha)$, where γ is a random variable drawn uniformly in [0,1]. Depending on the value of γ, β will therefore range from α to 1. Consequently, the size of small images $\alpha\beta$ takes values from α^2 to α, while large images will range from α to 1 (original, uncropped size). From here on, when mentioning β, we will be referencing this definition.

Until now we've only been talking about letting the Agent enlarge the frames and how this could be achieved, but we haven't discussed which component of the Agent specifically will be in charge of this task. We have the choice between the VAE and the RNN.

VAE

The VAE's goal is to complete clear shapes that extend beyond the frame's borders, like the street in the car racing game. We propose 2 methods for training the VAE:

- VAE (1): One encoder (E) and two decoders (D1, D2) are used [Ngu+19]. E and D1 are trained to encode and decode the original frames, while E and D2 are used to enlarge the small images, by encoding (n', m') and decoding (n, m). Finally, we use E and D2 to enlarge the original frames. The goal is for E to convert small and large images into similar latent code, so that decoder D2 can make sense of original frames latent code at prediction time.
- VAE (2): Using a encoder (E) and decoder (D), trained to encode small images into original images. At prediction time, original images are fed through this VAE as if it were small images to get extended frames. This model makes the assumption that the distribution of objects in original and small images is the same (for the Carracing game, there is always a road with green boundaries on the side), and the VAE just has to deal with scale.
- For both models, we experiment on both with and without β crop, both fixed and random α factor. After training, we also try expanding each image multiple times.

RNN

Unlike the VAE, the RNN can deal with temporal dependencies and will therefore also be able to keep track of elements that left the screen, like the monster and the fireball in Figure (1b). We propose 2 methods for training the RNN.

- RNN (1): We use a pretrained Model VAE (2) to enlarge the images. At each time frame, the RNN is trained to predict the code of next time frame,

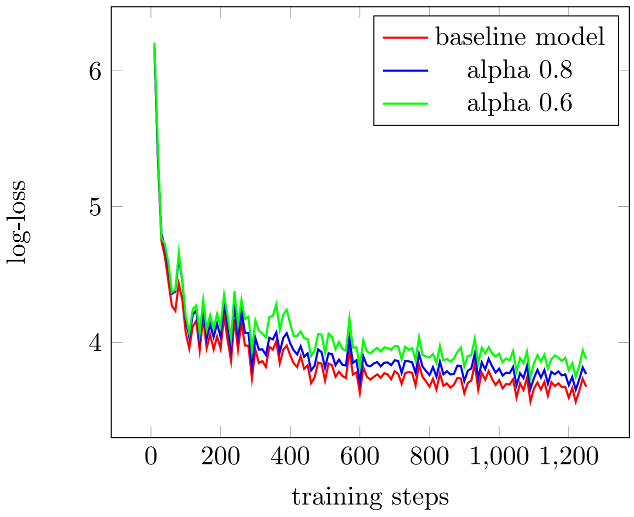


Figure 2: Influence of α on the training loss of the VAE

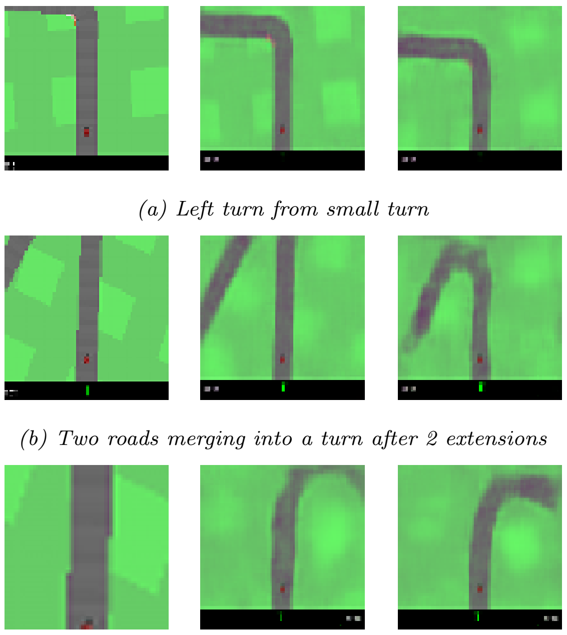
given the current frame's code, rewards and actions. At prediction time, we use the VAE again to reconstruct the images. In this model, the RNN contains temporal dependencies but is not in charge of enlarging the image.

- RNN (2): We use a traditional VAE, trained to encode all sorts of images (small and large ones). The RNN is trained to predict next original (large) latent code given current small images latent code. This model gives more responsibility to the RNN since it also has to produce enlarged frames, while the VAE only has to recontruct trivially.

Results

VAE Experiments[4]*:* We trained both VAE architectures on 10'000 rollouts with random policy on the Car Racing game. While both models converged to a low training loss, VAE Model (1) produced significantly blurrier and lower quality (sometimes even broken) images than VAE Model (2). This can be due to the fact that since in VAE Model (1), the original and small images share the latent space, they might have been mapped to different modes.
On the other hand, VAE Model (2) manages to generalize the extension to new images of original (non-cropped) size consistently, predicting full curves properly, as shown in Figure (3a), from a small curve in the road. We were also able to enlarge the extended frame multiple times to get an even larger scale image. By doing this, the model was surprisingly even able to understand that two parallel roads will converge into a curve later as in Figure (3b). It is also successfully able to extend lower scale images shown in Figure (3c). As could be expected, having a small α makes the task harder (see Figure 2), but allows to extend to a higher scale. Therefore, there is a trade-off between getting sharp images, and having a greater zooming effect.

[4]All our experiments can be found in `Report Results.ipynb`



(a) Left turn from small turn

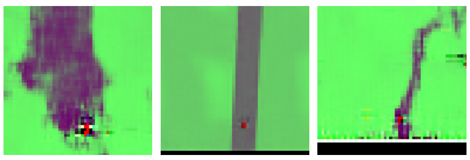
(b) Two roads merging into a turn after 2 extensions

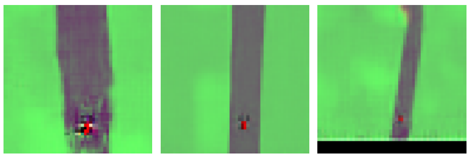
(c) Lower scale road with small turn extended to right turn

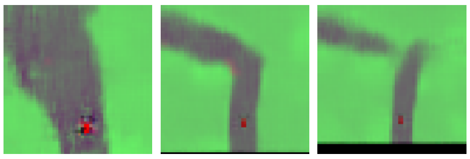
Figure 3: Original / Extended once / Extended twice images by the VAE, trained with α = 0.8

Image scaling in latent space: Scaling both images by a β factor at training time allows the VAE to not only learn a fixed output size, but is able to generate images augmented by α^{-i}, for any $i > 1$, from the same input image. This in turn implies that its latent space contains scaling information. Interestingly, by experimenting on changes in latent variables in the Car Racing environment, we found that one of its 32 latent features is responsible for scaling. It is possible to zoom in and out of the image by modifying the variable on position 1 in the range from -100 to 100, illustrated in Figure (4). We could therefore make the VAE (which was only trained to augment by α!) enlarge the input image by any amount and not be restrained to factors of α. While it makes the extended images blurrier, this shows that β introduces robustness to the model.

RNN experiments: We trained the MDN-RNN model on the Doom game, where temporal information and memorization is relevant. Our main concern was whether or not the RNN is capable of remembering fireballs and monsters that leave the screen. The answer is partly yes.
As mentioned before and illustrated in Figure (2), our α and β introduce additional difficulty for the VAE to reconstruct the images properly. As a consequence, the monsters are often not more than a very vague shadow in front of an already dark wall. The RNN therefore didn't consider them to be of great importance and doesn't keep track of them when they leave the screen. Indeed, it could be argued that the monsters' exact positions are not really important for the gameplay, since only the fireballs that they throw are a real threat to the agent. Figure (5) shows that both of our considered models for the RNN are successful at remembering those fireballs. They both keep track of a ball that dis-


(a) Straight road first sample, steer right afterwards


(b) Straight road second sample, no steering afterwards


(c) Steer left example

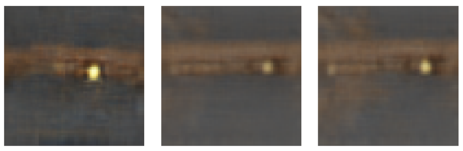
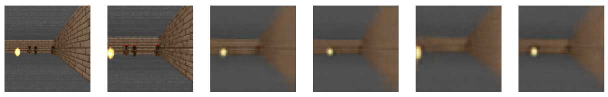
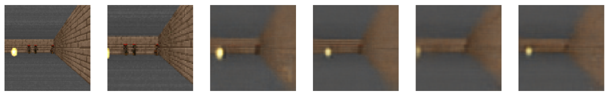
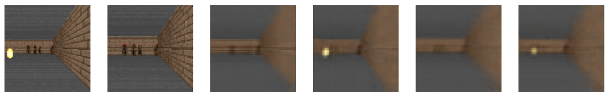
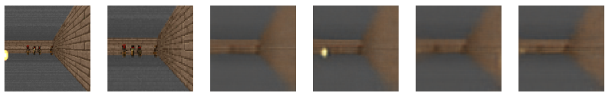

(d) Doom fireball view

Figure 4: Movement of latent variable on position 1 for Carracing on 4a, 4b, 4c, where we see zooming feature, from zoom-in on the left to the zoom-out on the right. On the Doom environment 4d, latent position that vas varied was on latent position 11.


(a) Fireball visible in input image


(b) Fireball disappearing in input image


(c) Fireball left the screen in input image, but RNNs still remember it


(d) Second frame in which fireball is remembered

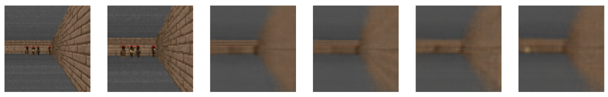

(e) Fireball left the screen of target image. RNN (2) still predicts some residual flares

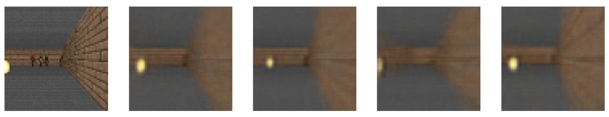
Figure 5: 5 successive frames:
Col 1: Original (target) image
Col 2: Input (cropped) image with $\alpha = 0.75$
Col 3: Prediction of image in col 2 by augmenting-VAE
Col 4: Prediction by simple RNN (1) with col 3 as input
Col 5: Prediction of image in col 2 by simple VAE
Col 6: Prediction by augmenting-RNN (2) with col 5 as input

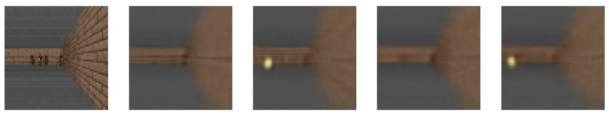
appeared from the input image two frames ago, until it also leaves the target image. RNN (2) is so eager to remember the fireball that it introduces residual flares into the image long after the ball disappeared. This is somewhat surprising, since we expected a relatively simple RNN (one LSTM with 256 hidden units) to be overwhelmed by the task of both augmenting an image and remembering objects at the same time. This is of course pure speculation, since the VAE and the RNN are trained separately and we never know, how easily the VAE lets the RNN include its temporal information into the latent space. Training the two models again might lead to one model being way superior to the other, and vice-versa.

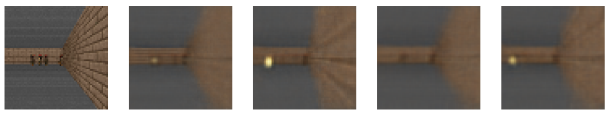
The previous example gets cropped images as input. So the input images have the same size as the images in the training set. How do the two models perform if they receive the original-size images as input, i.e. images of a size that they have never encountered during training?
Figure (6) shows that both models manage to remember the fireball for three more frames. In frame 4 after the fireball disappeared in the input, RNN (1) no longer predicts any signs of a fireball. This can have two reasons: either it predicted that the fireball left the border, or it is not accurate enough. Given that its model of the ball's trajectory is not very accurate, we tend to thinking that the second reason is more probable. In this respect, RNN (2) performs a lot better. When one looks very closely at the last images in Figures (6c) and (6d), one can see remainders of a fireball with a very accurate trajectory until it correctly leaves the screen. Therefore, we conclude that RNN (2) generalizes better than RNN (1) in this case.

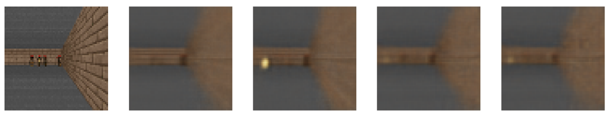
Discussion

The single VAE model produces coherent images, is sufficient for non time-dependent games, and can model scaling information, although it may not always generate good quality results. Extending multiple times successfully shows that it is robust to changes in scale. An interesting application would be to see how uncertain the controller is as a function of how far it sees. Concerning the RNN, the task of predicting the next enlarged frame is significantly harder than just predicting the next frame. Indeed, in the original model [HS18], since the game frames time steps are really close to each other, the model can make use of the high similarity between the current and next frame. The enlarging framework however has to construct elements that are not present in the current frame, and therefore necessarily needs to deal with temporal dependencies.
An idea for future work would be to train the VAE and RNN at the same time as a VAE-LSTM [Che+19].


(a) Fireball disappearing in input image


(b) Fireball left the screen in input image, but RNNs still remember it


(c) Second frame in which fireball is remembered


(d) Third frame in which fireball is remembered

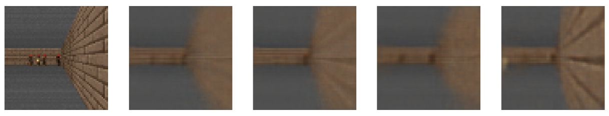

(e) RNN (1) does no longer show the fireball, while RNN (2) still includes on the very left edge

Figure 6: 5 successive frames:
Col 1: Input (original-size) image
Col 2: Prediction of image in col 1 by augmenting-VAE
Col 3: Prediction by simple RNN (1) with col 2 as input
Col 4: Prediction of image in col 1 by simple VAE
Col 5: Prediction by augmenting-RNN (2) with col 4 as input

In our use case, this might lead to better performance since the VAE would divide the latent space in such a way as to make it easy for the RNN to include the temporal information.
One could also not restrict the model to predict the surrounding of the image, but let it choose which region to extend or to discover, using some attention mechanism for instance. In some sense, this would create an "omnipresent" agent.

Summary

We have presented a way to do spatial prediction, based either on an extending VAE for tasks where all information is contained in current observation, or on temporal representations learned by an RNN. Therefore, the RNN allows to predict objects that were seen before disappearing and that may reappear at some point. We also found that the VAE encodes zoom information in its latent space, and is able to extend multiple times images of different scales. Those two concepts combined give insights for new type of predictive modelling, based on both spatial coherence and temporal information.

References

[Sch90a] J. Schmidhuber. “An on-line algorithm for dynamic reinforcement learning and planning in reactive environments”. In: (June 1990), 253–258 vol.2. ISSN: null. DOI: 10.1109/IJCNN.1990.137723.

[Sch90b] Jürgen Schmidhuber. “Making the World Differentiable: On Using Self-Supervised Fully Recurrent Neural Networks for Dynamic Reinforcement Learning and Planning in Non-Stationary Environments”. In: (1990).

[MW91] J. Meyer and S. W. Wilson. “A Possibility for Implementing Curiosity and Boredom in Model-Building Neural Controllers”. In: (1991), pp. 222–227. ISSN: null. URL: https://ieeexplore.ieee.org/document/6294131.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[KW13] Diederik P Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: (2013). arXiv: 1312.6114 [stat.ML].

[Bro+16] Greg Brockman et al. *OpenAI Gym*. 2016. eprint: arXiv:1606.01540.

[PN17] Athanasios S. Polydoros and Lazaros Nalpantidis. “Survey of Model-Based Reinforcement Learning: Applications on Robotics”. In: *Journal of Intelligent & Robotic Systems* 86.2 (May 2017), pp. 153–173. ISSN: 1573-0409. DOI: 10.1007/s10846-017-0468-y. URL: https://doi.org/10.1007/s10846-017-0468-y.

[HS18] David Ha and Jürgen Schmidhuber. “World Models”. In: *CoRR* abs/1803.10122 (2018). arXiv: 1803.10122. URL: http://arxiv.org/abs/1803.10122.

[har18] hardmaru. *WorldModelsExperiments*. https://github.com/hardmaru/WorldModelsExperiments. 2018.

[Che+19] Run-Qing Chen et al. “Sequential VAE-LSTM for Anomaly Detection on Time Series”. In: *arXiv e-prints*, arXiv:1910.03818 (Oct. 2019), arXiv:1910.03818. arXiv: 1910.03818 [cs.LG].

[Dav19] DavidADSP. *AppliedDataSciencePartners / WorldModels*. https://github.com/AppliedDataSciencePartners/WorldModels. 2019.

[FMH19] C. Daniel Freeman, Luke Metz, and David Ha. “Learning to Predict Without Looking Ahead: World Models Without Forward Prediction”. In: (2019). arXiv: 1910.13038 [cs.NE].

[Hak19] Simon Hakenes. *OpenAI Gym, Vizdoom gym wraper*. https://github.com/shakenes/vizdoomgym. 2019.

[Ngu+19] Thanh Thi Nguyen et al. “Deep Learning for Deepfakes Creation and Detection”. In: (2019). eprint: arXiv:1909.11573 (cs.CV).

[RS19] Sebastian Risi and Kenneth O. Stanley. “Deep Neuroevolution of Recurrent and Discrete World Models”. In: *CoRR* abs/1906.08857 (2019). arXiv: 1906.08857. URL: http://arxiv.org/abs/1906.08857.