
How do Mixture Density RNNs Predict the Future?

Kai Olav Ellefsen¹ Charles Patrick Martin^{1,2} Jim Torresen^{1,2}

Abstract

Gaining a better understanding of how and what machine learning systems learn is important to increase confidence in their decisions and catalyze further research. In this paper, we analyze the predictions made by a specific type of recurrent neural network, mixture density RNNs (MD-RNNs). These networks learn to model predictions as a combination of multiple Gaussian distributions, making them particularly interesting for problems where a sequence of inputs may lead to several distinct future possibilities. An example is learning internal models of an environment, where different events may or may not occur, but where the average over different events is not meaningful. By analyzing the predictions made by trained MD-RNNs, we find that their different Gaussian components have two complementary roles: 1) Separately modeling different stochastic events and 2) Separately modeling scenarios governed by different rules. These findings increase our understanding of what is learned by predictive MD-RNNs, and open up new research directions for further understanding how we can benefit from their self-organizing model decomposition.

1. Introduction

Deep learning has greatly increased the ability for computers to perform complex tasks from a wide range of domains, including image recognition, language modeling, game playing and predicting the future (Mnih et al., 2015; LeCun et al., 2015; Wichers et al., 2018). However, we have an incomplete understanding of exactly how the deep learning models learn to perform these tasks. Gaining a better understanding of these models is considered to be one of the most important current challenges in artificial intelligence (Samek et al., 2017; Garcia et al., 2018). There are many reasons why a better understanding is important, ranging from increasing

the ability to trust machine learning systems, to the benefits such understanding would have for continued research and development of algorithms. There are therefore many recent studies aiming to make deep learning models more understandable and explainable, for both convolutional neural networks (CNNs) (Yosinski et al., 2015), recurrent neural networks (RNNs) (Karpathy et al., 2015) and other architectures (Smilkov et al., 2017).

In this paper, we aim to gain a better understanding of one specific neural network architecture, mixture density RNNs (MD-RNNs). MD-RNNs are recurrent neural networks combined with a mixture density network (Bishop, 1994; Bishop & Others, 1995), such that that the output parametrizes a mixture of Gaussians distribution (Figure 1). MD-RNNs are particularly interesting for tasks involving creative prediction, since the recurrent part allows the modeling and forecasting of sequences, and the Gaussian mixture part allows predictions to be creative, modeling different types of scenarios in a single neural network (Ha & Eck, 2017).

MD-RNNs recently gained significant attention (Pearson, 2018; Yao, 2018) in Ha and Schmidhuber’s paper on “World Models” (Ha & Schmidhuber, 2018), which demonstrated that MD-RNNs, can learn to predict the future from a large number of observations of a simulated world. The internal models learned in Ha and Schmidhuber’s work represent the agent’s world so well that the authors were able to 1) train an agent *inside its own internal model* (or, said differently, inside its own “dream”) and 2) to be the first agent to solve the Car Racing environment in OpenAI gym. Despite the impressive results and the wide attention this work has gotten, we do not have a good understanding of how predictive MD-RNNs model the world.

MD-RNNs make predictions by sampling from a probability distribution with multiple different sub-distributions. We investigate two hypotheses about the role of these sub-distributions (mixture components) when MD-RNNs predict the future. The hypotheses are: 1) Different mixture components model different stochastic events and 2) Different mixture components model different situations with different “rules” (that is, different internal models – Figure 1). We train world models in a Doom game environment, similarly to (Ha & Schmidhuber, 2018), and let them hallucinate imagined scenarios. From these scenarios, we extract events

¹Department of Informatics, University of Oslo, Norway
²RITMO, University of Oslo, Norway. Correspondence to: Kai Olav Ellefsen <uio>.

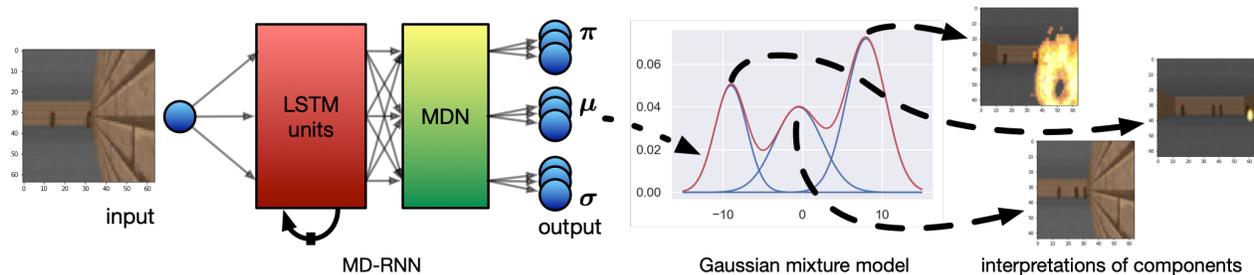


Figure 1. Left: MD-RNNs model data with probability distributions composed of several components, parametrized by π , μ and σ . Right: We investigate the roles of individual components to gain a better understanding of how MD-RNNs make predictions. One possibility (illustrated here) is that different mixture components represent situations governed by different rules.

and situations according to our two hypotheses, and then measure to which degree different events are produced by different components of the mixture model.

The main contributions of this paper are 1) A framework for automatically measuring the tendency for different components of a Gaussian mixture model to generate particular types of prediction, and 2) New insights into the roles of the Gaussian components of trained MD-RNNs. In particular, we find evidence for both our hypotheses, including a very clear demonstration of different mixture components self-organizing to serve as internal models for scenarios with different rules.

2. Background

2.1. MD-RNNs

Generative machine learning models for content such as text, images or sound typically model the generated content with a probability distribution (Goodfellow et al., 2016). Mixture density networks (MDNs) are neural networks that represent mixture density models (McLachlan & Basford, 1988), that is, probability distributions which are composed of several sub-distributions (several Gaussian distributions in the models applied here – see Figure 1). MDNs can in principle represent any conditional probability distribution, and are useful when the modeled phenomenon is not well represented by a simpler distribution. An example, which we study here, is learning internal models of an environment, where different events may or may not occur, but where the average over different events is not meaningful. In this case, the multiple sub-distributions of a mixture density model can help model the fact that the world has multiple possible states which should not be mixed together or averaged.

In practice, a mixture density network (MDN) operates by transforming the outputs of a neural network to form the parameters of a mixture distribution (Bishop, 1994), generally with Gaussian models for each mixture component. These

parameters are the centres (μ) and scales (σ) for each Gaussian component, as well as a weight (π) for each component (see Figure 1). The MDN usually uses an exponential activation function to transform the scale parameters to be positive and non-zero. For training, the probability density function of the mixture model is used to generate the negative log likelihood for the loss function. This involves constructing probability density functions (PDFs) for each Gaussian component and categorical distribution from the mixture weights (see Appendix Section 1.4 for details). One advantage of an MDN is that various component distributions can be used so long as the PDF is tractable, for instance, 1D (Bishop, 1994) or 2D (Graves, 2013) Gaussian distributions, or, as in our case, a multivariate Gaussian with a diagonal covariance matrix.

For inference, results are sampled from the mixture distribution. First, the π s are used to form a categorical distribution by applying the softmax function. A sample is drawn from this distribution to determine which Gaussian component will provide the output. The index i of the sampled π is used to select a Gaussian distribution, $\mathcal{N}(\mu_i, \sigma_i^2)$, from which a sample is drawn to provide the outcome. In some cases, it is advantageous to adjust the diversity of sampling (for instance, to favour unlikely predictions), in which case the temperature of the categorical distribution can be adjusted in the typical way, and the covariance matrices of the Gaussian components may be scaled. We refer these operations as adjusting π - or σ -temperature respectively.

An MDN can be applied to the outputs of an RNN, forming an MD-RNN. This approach has been applied to model 2D pen data, such as for handwriting (Graves, 2013) and sketches (Ha & Eck, 2017) as well as musical performance (Martin & Torresen, 2018). Other applications include parametric speech synthesis (Wang et al., 2017), and identifying salient locations in video data (Bazzani et al., 2017). Ha and Schmidhuber applied an MD-RNN model to model the future state of a video game screen image and as-

sist an RL agent (Ha & Schmidhuber, 2018). In the present research, we delve into this application to understand what such a model learns about the virtual worlds and how this information is represented.

2.2. Predicting the future with deep neural networks

Progress in deep learning has recently made it possible to learn to predict future frames of video from observing sequences of video frames (Finn et al., 2016; Mathieu et al., 2016). However, most approaches for predicting future visual input from pixels have typically only had the ability to predict a few frames into the future before predicted images get blurry or static. Recently, techniques have been developed that attempt to mitigate this limitation by first encoding frames into a compact, high-level representation, then predicting how this compact representation develops over time. Finally, decoding the predicted compact representation produces a predicted future image. Compared to predictions made directly in pixel-space, such high-level predictions degrade less quickly, demonstrating good prediction performance many seconds into the future (Villegas et al., 2017; Wichers et al., 2018; Ha & Schmidhuber, 2018).

In Ha and Schmidhuber’s recurrent world model (Ha & Schmidhuber, 2018), the predictive model consists of two components: 1) A visual component (V), which learns an encoding/decoding between a visual scene and a compact representation and 2) A memory component (M), which learns how the compact representation develops over time (Figure 2). The first component is learned by a variational autoencoder (VAE (Kingma & Welling, 2013)), by presenting it with a large collection of pictures from the visual scene. The second is learned by an MD-RNN. This world model was demonstrated to be able to predict many frames into the future, and in fact to “dream” whole episodes of agent experience. It is, however, not clear what role the different mixture components in the MD-RNN play in predicting the future.

2.3. Architectures for multiple internal models

One of our hypotheses suggests that the different Gaussian components learn to model different situations with different “rules”, that is, situations where predictions need to be so different that they are best modeled separately. Humans show a remarkable ability to learn internal models (mental simulations) of a wide range of different situations, objects and people, without a high degree of conflict or interference between them.

One theory suggests that this ability is facilitated by the modular organization of our central nervous system. Neural modularity may be a key to allow multiple internal models to coexist, enabling the selection of the appropriate actions for the current context (Ghahramani & Wolpert, 1997; Wolpert

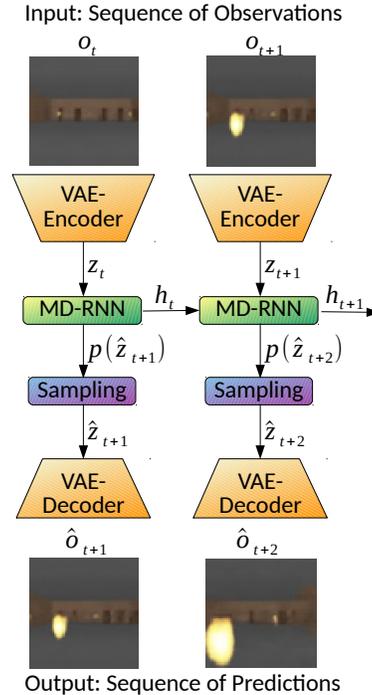


Figure 2. World model predicting future frames by combining a variational autoencoder and an MD-RNN. We follow the architecture suggested in (Ha & Schmidhuber, 2018).

et al., 2003). Computational models built around this idea have indeed demonstrated the ability to learn and maintain multiple internal models, and select the appropriate model for a given context (Wolpert & Kawato, 1998; Haruno et al., 2001; Demiris & Khadhouri, 2006). These models work by dividing learning experiences into multiple modules, where different modules *compete* to represent different situations. After a number of learning episodes, this causes different modules to specialize at representing different internal models, allowing the system to model situations with different rules, with minimal interference. Our hypothesis suggests that the Gaussian components of the MDN self-organize to perform a similar task, allowing scenarios with different rules to be modeled with little interference.

3. Methods

3.1. World Model MD-RNN

Ha and Schmidhuber’s world model (Ha & Schmidhuber, 2018) combines an MD-RNN and a VAE to predict future states of a video game screen (Figure 2). By training the VAE to compress representations of visual scenes, the MD-RNN has a more manageable job of predicting how scenes unfold in the future.

Training the model happens in two steps: First, the VAE is trained on examples of images from the environment in which we wish to learn to make predictions. The VAE compresses each image (64x64 pixels, with 3 color channels in our setup) into a latent vector, z (64 floating-point numbers in our case). It then attempts to reconstruct the same image from the latent vector. The VAE is trained to both reconstruct images as well as possible, and to keep the representations of similar inputs close together in latent space (details are found in Appendix Section 1.3). This allows small changes to the latent vector to give meaningful changes in the compressed images.

After the VAE has learned to compress images of the world into latent vectors, the MD-RNN can be trained on sequences of latent vectors. We follow (Ha & Schmidhuber, 2018) in applying a single-layer LSTM (Hochreiter & Schmidhuber, 1997), trained by seeing examples of sequences of images as input, and the same sequence, shifted by one time-step, as outputs. Thereby, the LSTM learns to predict the next latent vector from a sequence of previous observations. More details on the World Model MD-RNN are found in the Appendix Section 1.4.

3.1.1. DATA COLLECTION AND TRAINING

The data collection and training process follows (Ha & Schmidhuber, 2018), except we do not train a controller, since we are here *analyzing* predictions, and not using them for agent control. The process can be summarized in the following steps (more details in Appendix Section 1¹).

1. Simulate 2,000 episodes with a random policy. Store all actions taken and frames observed.
2. Train a VAE to encode each frame into a length 64 latent vector z , and to decode z back to the same image.
3. Generate latent vectors z for each frame from the simulated episodes. Further training can now be done without the actual images.
4. Train an MD-RNN to model $P(z_{t+1}|a_t, z_t, h_t)$, that is, the probability distribution for next latent vector, given the current latent vector and action, as well as the RNN’s hidden state.

3.2. Training Scenario

We follow (Ha & Schmidhuber, 2018) in training the predictive MDN-RNNs to model the VizDoom (Kempka et al., 2017) Take Cover scenario². This scenario takes place in a

¹Experiment code is available at: <http://doi.org/10.5281/zenodo.2539145>

²<https://gym.openai.com/envs/DoomTakeCover-v0/>

rectangular room, where a player is facing monsters on an opposite wall. Monsters will fire exploding fireballs at the player, and the player attempts to survive as long as possible by moving left and right, dodging the incoming projectiles. Agents receive 3D images of the scene ahead of them as input, and make only one decision at each timestep: Move to the left, move to the right or stay in the same place.

This scenario serves as a useful test for our hypotheses, since it has both stochastic events (e.g., monsters may or may not launch fireballs) and different situations governed by different rules (an exploding fireball behaves very differently than an incoming fireball mid-air).

3.3. Measuring how MD-RNNs make predictions

After training MD-RNNs, we analyze the predictions they make when “dreaming” about the future. We insert an initial latent vector (representing the real initial state from the game) into the MD-RNN, and then repeat the steps below for as long as we want to predict (Figure 3 illustrates the steps, following the same numbering as the list):

1. Produce a probability distribution over the next latent vector, $P(\hat{z}_{t+1}|a_t, \hat{z}_t, h_t)$ parametrized by the MDN-parameters, π, μ and σ . Store π (the vector indicating the weight of each Gaussian component).
2. Sample a latent vector \hat{z}_{t+1} from the probability distribution, and decode it into a predicted frame with the VAE.
3. Analyze the predicted frame to measure which events are depicted (see below). Store the list of events for the current frame together with π . Together, these can tell us whether different mixture components generate different events.
4. Repeat the process, starting from point 1, with the sampled \hat{z}_{t+1} as the RNN input, to predict the next latent vector $P(\hat{z}_{t+2}|a_{t+1}, \hat{z}_{t+1}, h_{t+1})$

Every round through this process generates a new predicted latent vector, which is next used as input to predict the latent vector following it. Storing latent vectors and MD-RNN parameters allows us to subsequently analyze the way the MD-RNN has learned to represent the world and make predictions about it.

3.3.1. MEASURING EVENTS IN PREDICTED FRAMES

Our hypotheses suggest that different mixture components represent either different stochastic events, or different situations where different rules apply. In the world we make predictions about, we identify two stochastic events: 1) Monsters may appear, and 2) they may launch a fireball

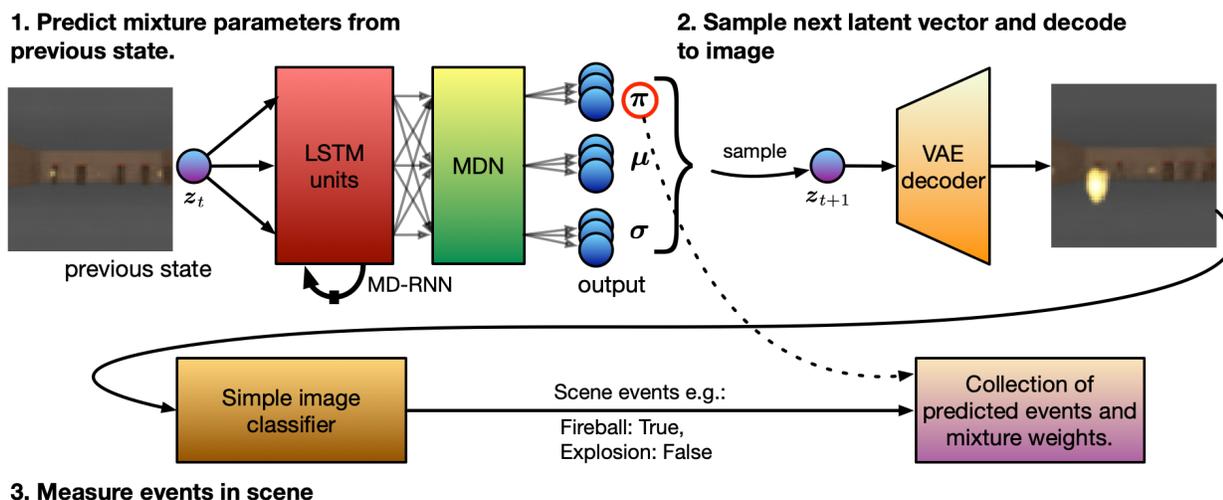


Figure 3. Our proposed framework for analyzing how MD-RNNs make predictions

towards the player. Note that monsters never disappear in the modeled world, and fireballs disappearing is not a stochastic event, since once a fireball has been fired, it will deterministically disappear after reaching the other end of the room.

For our second hypothesis, we identify three situations where the rules for how frames evolve in a time sequence are very different: 1) The normal situation (player is facing monsters, who sometimes launch fireballs), 2) an explosion takes place in front of the player and 3) the player is next to a wall. Situation 2 and 3 are so different from the normal situation that internal models of the three different situations could benefit from some separation. Explosions cover a large portion of the screen, and unfold according to a specific sequence, which has little to do with the way a normal scene unfolds (see Figure 5). Walls next to the agent result in unique dynamics, since they require a large portion of the screen to move sideways (in the opposite direction) as the player moves.

Since we are dealing with a quite simple and limited world, we can measure events from frames with straightforward image processing methods from the Python package scikit-image³. The methods we apply to measure the presence of monsters, fireballs, walls and explosions are documented in Appendix Section 2, and also made available online⁴.

³<https://scikit-image.org/>

⁴<http://doi.org/10.5281/zenodo.2539145>

4. Results

As previously discussed, we have two main hypotheses about the roles of different mixture components in the MD-RNN: 1) Different components learn to model different possible futures, allowing them to creatively sample what will happen next, and 2) Different components learn to form different *internal models* of the environment, that is, they specialize to model situations governed by a specific set of rules. Below, we analyze MD-RNN predictions along with the weights of mixture components to shed light on these hypotheses.

4.1. Common parameters

In our main experiments, we test 5 independently trained MD-RNNs, all with the same architecture (see Appendix Section 1), to reduce the chance that results are specific to one trained model. In practice, we found results to be very similar when training the same model multiple times with shuffled data. For all five, we generate multiple “dreams” by predicting future latent vectors, and feeding each prediction in as the input-vector to the RNN for the next time-step, along with a randomly sampled action. This allows us to dream up long prediction sequences which, although not always realistic, illuminate how mixture components relate to predicted events. In our main experiments, we generate 10 dreams for each of our 5 models, each dream 1000 time-steps long. Tests of statistical significance apply the Mann-Whitney U test.

4.2. Analyzing frames produced in prediction sequences

4.2.1. STOCHASTIC EVENTS

Our first hypothesis suggests that different mixture components represent different stochastic events, allowing creative predictions about the future by sampling from different Gaussian components. We test this hypothesis by dreaming up many different futures as described above, and measuring 1) different *stochastic events* in the dreams and 2) the weight assigned to each component in the mixture model. As mentioned above, there are two different stochastic events in this scenario: fireballs appearing and monsters appearing.

To confidently say that a specific mixture component is particularly responsible for producing one event, we need to measure whether that component has produced the event more frequently than one would expect if events were evenly distributed among components. For instance, if we find that one component is responsible for 80% of the fireball appearances, but that component is also responsible for 80% of all generated frames, then we do not have any clear evidence. We therefore measure the relationship between components and events as follows:

1. Produce 10 different dreams with each of the 5 trained MD-RNNs, resulting in a total of 50 dreams.
2. For each time-step of a dream, measure a) the presence of the events described above, and b) which component is currently the most active (the one with the highest π -value output by the MD-RNN).
3. Within one dream, the component that produced an event most frequently, is denoted as the “main component” for that event. This is the Gaussian that is most likely responsible for generating the given event.
4. Measuring the proportion of the event produced by the “main component” versus the other components across all N dreams yields the leftmost boxes in the pairs in Figure 4.
5. To be sure the “main component” is specifically responsible for the specific event/situation, we also measure the proportion of *all* frames produced by that component. This yields the rightmost boxes.
6. A significantly higher value in the leftmost than the rightmost box thus indicates that one component is producing the relevant event/situation more frequently than one would expect by looking at the proportion of all events generated by that component.

As we can see in Figure 4, there is a strong tendency for fireball appearances to be produced more by one specific

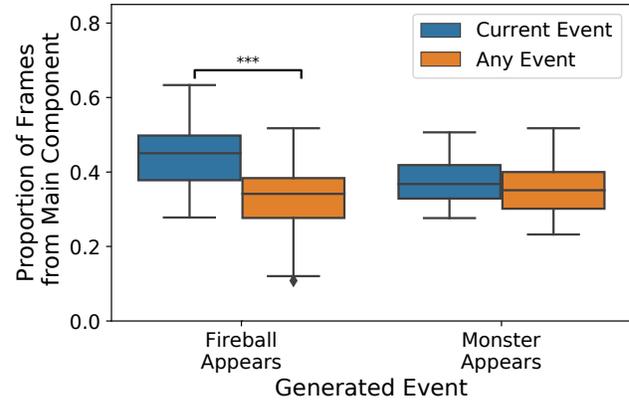


Figure 4. The tendency for different stochastic events to be produced by one specific Gaussian component (blue) vs the tendency for that component to be responsible for events overall (orange). *** indicates significant differences with $p < 0.001$.

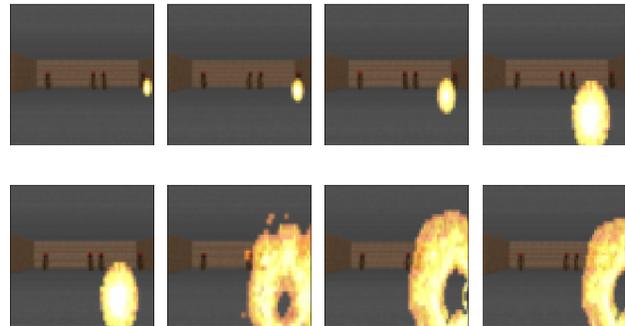


Figure 5. Top: A monster launching a fireball at the player. Bottom: An explosion unfolding in front of the player. The two situations are governed by very different rules. Images down-sampled to the same resolution (64x64) used during training.

component. There is no similar tendency for monster appearances.

4.2.2. DIFFERENT INTERNAL MODELS

Our second hypothesis is that different mixture components represent different *internal models*, that is, models of scenarios where the rules are different. To study this, we repeated the calculations outlined above, measuring the presence of such scenarios, rather than stochastic events. As discussed above, we identify 3 scenarios in this game where the rules of how to generate the next frame are very different from the normal situation (facing monsters and any fireballs): 1) having a wall on the left, 2) having a wall on the right and 3) getting hit by an exploding fireball.

The result of this calculation is shown in Figure 6. There are statistically significant differences ($p < 0.001$) between the main component’s tendency to generate the specific situations and their tendency to generate frames overall, for explosions and walls on either side. We also show that the same effect is not generally present for situations containing fireballs. We hypothesize that this is because fireballs are very common, and do not drastically change the way the world changes from one frame to the next. There should therefore be less need for modeling them in a separate mixture component.

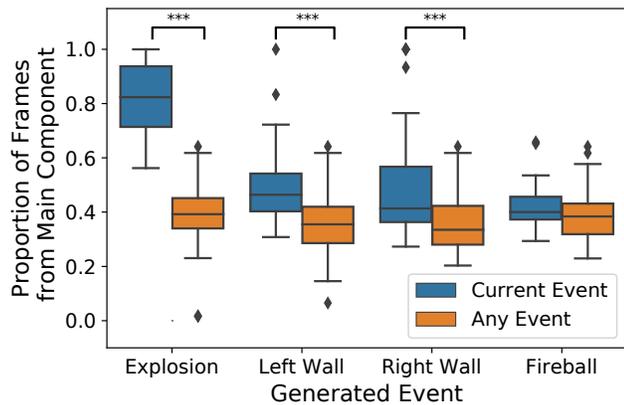


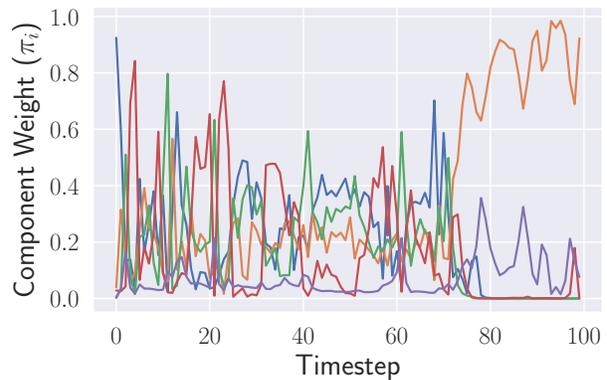
Figure 6. The tendency for different scenarios to be produced by one specific Gaussian component (blue) vs the tendency for that component to be responsible for events overall (orange). * * * indicates significant differences with $p < 0.001$.

4.3. Plotting component weights and dreams

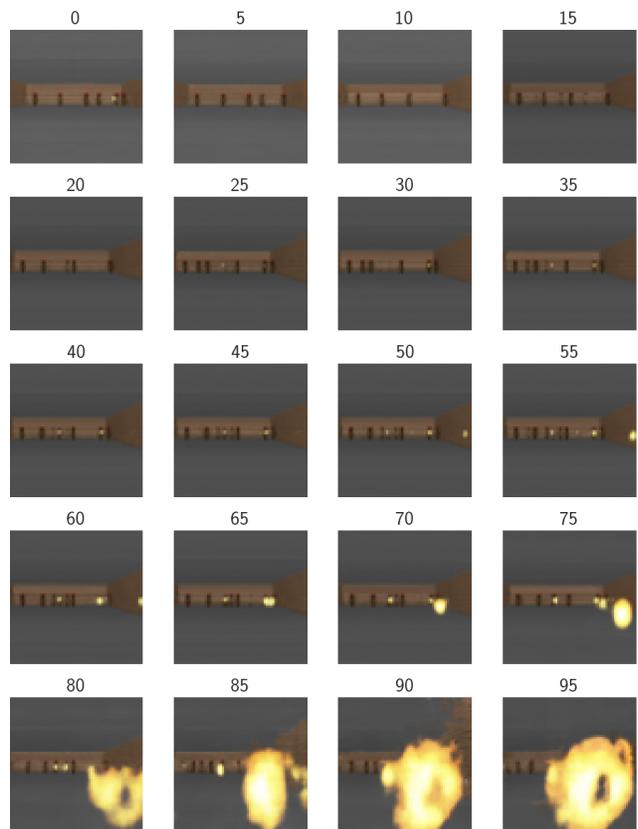
To further illuminate the role of different mixture components, we let a single trained MD-RNN dream up 100-timestep predictions, while plotting the weights of all components, to see which one is currently most responsible for making predictions. An example of such a plot is shown in Figure 7. Notice one specific mixture component dominates from around timestep 75, the same time that an explosion is present in the frame. In other repetitions of the experiment, we found a different component tends to dominate when the agent is near a wall. In “normal” situations (no nearby walls or explosions), we tend to see different components being active together, without any clear dominance. This supports our second hypothesis, that different components specialize to model scenarios where the rules for generating future frames are different.

4.4. Committing to one mixture

As a final test of the role of different mixture components in making predictions, we generated dreams while *committing* to a single component during an entire dream. We condi-



(a) The weights of the 5 different mixture components (π_i for $i \in \{1, 2, 3, 4, 5\}$) during a 100-timestep dream.



(b) The resulting dream generated by sampling according to the mixture weights in (a). Numbers indicate the current timestep.

Figure 7. The weights over time of each of the 5 mixture components output by the MD-RNN and the corresponding dreams produced by sampling according to those weights. Until timestep 75, several components are similarly weighted, and responsible for making predictions together. After timestep 75, one component dominates, and takes over in generating predictions. The resulting prediction is an exploding fireball.

tioned the MD-RNN with a random start image, and made a dream predicting 1000 steps into the future, sampling only from the *first mixture component*. We then repeated this for each of the five mixture components in the MD-RNN. Since different trained models may not represent the same events in the exact same mixture components, we base this analysis on ten 1000-timestep dreams for each component, from a single trained model.

The results are shown in Figure 8. There is a clear tendency for this model to generate explosions with the second mixture component, and walls (both right and left) with the fifth component. Visualizing the conditional dreams, we observe something interesting: The components that do not produce explosions result in dreams where fireballs approach the player, but stop and hover mid-air, or even reverse and return to the monsters. Presumably, these components have never learned to model explosions, and can therefore not produce them when being responsible for generating dreams alone.

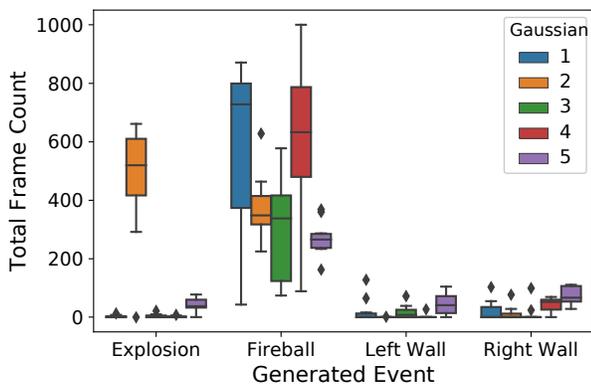


Figure 8. The events generated in dreams, when committing to a single Gaussian component during the entire prediction sequence.

5. Discussion

The results give support for both our initial hypotheses: The different Gaussian components in the MD-RNN specialize to model both different stochastic events (Figure 4) and different internal models (Figure 6). For the stochastic events, we saw a strong tendency for fireball appearances to be generated more frequently by a specific mixture component, but the same was not true for monster appearances. Detecting monsters in the generated predictions is more difficult than the other elements we detect (see Appendix section 2), and we cannot rule out that a relationship exists here that we could have measured if monster appearances were less ambiguous.

For the second hypothesis, we observed very clear evidence

that situations where the rules governing predictions are different were produced by separate Gaussian components. This was seen clearly both when measuring which components were most active in the different situations (Figures 6 and 7), and when sampling from a single component during an entire prediction sequence (Figure 8).

6. Conclusion

Through automatic classification of predicted frames, we have shed light on the way mixture density RNNs predict the future. We started out with two hypotheses for the role of the different components in the Gaussian mixture models, and found some evidence in support of both. First, we found evidence that different components produce different stochastic events more frequently, supporting the hypothesis that different components of the mixture models represent different potential directions for the predicted future. This is a valuable property for systems modeling creative predictions (such as in generation of artistic text, music and images), since it allows them to strike a balance between modeling an observed phenomenon and improvising by choosing between several possible predicted futures.

We found even more solid evidence for our second hypothesis: There is a very strong tendency for different components of the mixture model to be responsible for producing events that are governed by different “rules”, that is, events that require different internal models. Building machine learning systems that can represent different internal models is a long-standing challenge, since learning of very different skills tends to cause interference or forgetting (Ellefsen et al., 2015). One way this challenge has been handled in the past, is by building modular systems, where different modules *compete* to represent different internal models (Demiris & Khadhoury, 2006; Haruno et al., 2001). Our results suggest that mixture density RNNs self-organize to separate different internal models into different components.

This ability of MD-RNNs opens up a further hypothesis: Since these networks can automatically self-organize multiple internal models, they should be well equipped to model different scenarios with a low degree of interference. In future studies, we plan to examine this further by training MD-RNNs on multiple different environments, studying the effect of the number of components in the MDN on the observed interference.

7. Acknowledgments

This work is partially supported by The Research Council of Norway as a part of the Engineering Predictability with Embodied Cognition (EPEC) project, under grant agreement 240862.

References

- Bazzani, L., Larochelle, H., and Torresani, L. Recurrent Mixture Density Network for Spatiotemporal Visual Attention. In *International Conference on Learning Representations*, pp. arXiv:1603.08199, 2017.
- Bishop, C. M. Mixture density networks. Technical Report NCRG/97/004, Neural Computing Research Group, Aston University, 1994.
- Bishop, C. M. and Others. *Neural networks for pattern recognition*. Oxford university press, 1995.
- Demiris, Y. and Khadhour, B. Hierarchical attentive multiple models for execution and recognition of actions. *Robotics and Autonomous Systems*, 54(5):361–369, 2006. ISSN 09218890. doi: 10.1016/j.robot.2006.02.003.
- Ellefsen, K. O., Mouret, J.-B., and Clune, J. Neural Modularity Helps Organisms Evolve to Learn New Skills without Forgetting Old Skills. *PLOS Computational Biology*, 11(4):e1004128, 2015. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1004128. URL <http://dx.plos.org/10.1371/journal.pcbi.1004128>.
- Finn, C., Goodfellow, I., and Levine, S. Unsupervised Learning for Physical Interaction through Video Prediction. *ArXiv e-prints*, pp. arXiv:1605.07157, 2016.
- Garcia, R., Telea, A. C., da Silva, B. C., Tørresen, J., and Comba, J. L. D. A task-and-technique centered survey on visual analytics for deep learning model engineering. *Computers & Graphics*, 77:30–49, 2018.
- Ghahramani, Z. and Wolpert, D. M. Modular decomposition in visuomotor learning. *Nature*, 386(6623):392–395, 1997. ISSN 00280836. doi: 10.1038/386392a0.
- Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. Deep generative models. In *Deep learning*, chapter 20, pp. 651–716. MIT press Cambridge, 2016.
- Graves, A. Generating Sequences With Recurrent Neural Networks. *ArXiv e-prints*, 1308.0850, aug 2013. URL <https://arxiv.org/abs/1308.0850>.
- Ha, D. and Eck, D. A Neural Representation of Sketch Drawings. *arXiv e-prints*, 1704.03477, apr 2017. URL <http://arxiv.org/abs/1704.03477>.
- Ha, D. and Schmidhuber, J. Recurrent World Models Facilitate Policy Evolution. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 2451–2463. Curran Associates, Inc., 2018. URL <http://papers.nips.cc/paper/7512-recurrent-world-models-facilitate-policy-evolution.pdf>.
- Haruno, M., Wolpert, D. M., and Kawato, M. MOSAIC Model for Sensorimotor Learning and Control. *Neural Computation*, 2220:2201–2220, 2001.
- Hochreiter, S. and Schmidhuber, J. Long Short-Term Memory. *Neural Computation*, 1997. ISSN 08997667. doi: 10.1162/neco.1997.9.8.1735.
- Karpathy, A., Johnson, J., and Fei-Fei, L. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.
- Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaskowski, W. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games, CIG*, 2017. ISBN 9781509018833. doi: 10.1109/CIG.2016.7860433.
- Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature*, 521(7553):436–444, 2015. doi: 10.1038/nature14539.
- Martin, C. P. and Torresen, J. RoboJam: A musical mixture density network for collaborative touchscreen interaction. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10783 LNCS, pp. 161–176. Springer, Cham, 2018. ISBN 9783319775821. doi: 10.1007/978-3-319-77583-8_11. URL http://link.springer.com/10.1007/978-3-319-77583-8_{_}11.
- Mathieu, M., Couprie, C., and LeCun, Y. Deep multi-scale video prediction beyond mean square error. In *ICLR*, 2016.
- McLachlan, G. J. and Basford, K. E. *Mixture models: Inference and applications to clustering*, volume 84. Marcel Dekker, 1988.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. a., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015. doi: 10.1038/nature14236.
- Pearson, J. Watch a Computer Learn to Play Doom’ Inside a Dream [online article]. *Motherboard*, 2018. Retrieved from https://motherboard.vice.com/en_us/article/43bxjj/watch-deep-learning-ai-computer-play-doom-dream.

- Samek, W., Wiegand, T., and Müller, K.-R. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*, 2017.
- Smilkov, D., Carter, S., Sculley, D., Viégas, F. B., and Wattenberg, M. Direct-manipulation visualization of deep networks. *arXiv preprint arXiv:1708.03788*, 2017.
- Villegas, R., Yang, J., Zou, Y., Sohn, S., Lin, X., and Lee, H. Learning to Generate Long-term Future via Hierarchical Prediction. *ICML*, apr 2017.
- Wang, X., Takaki, S., and Yamagishi, J. An autoregressive recurrent mixture density network for parametric speech synthesis. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, mar 2017. doi: 10.1109/icassp.2017.7953087. URL <https://doi.org/10.1109/icassp.2017.7953087>.
- Wichers, N., Villegas, R., Erhan, D., and Lee, H. Hierarchical Long-term Video Prediction without Supervision, jul 2018. ISSN 1938-7228. URL <http://proceedings.mlr.press/v80/wichers18a.html>.
- Wolpert, D. M. and Kawato, M. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11(7):1317–1329, 1998. ISSN 08936080. doi: 10.1016/S0893-6080(98)00066-5.
- Wolpert, D. M., Doya, K., and Kawato, M. A unifying computational framework for motor control and social interaction. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 358(1431):593–602, 2003. ISSN 09628436. doi: 10.1098/rstb.2002.1238. URL <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1693134&tool=pmcentrez&rendertype=abstract>.
- Yao, M. No Time to Read AI Research? We Summarized Top 2018 Papers for You [Blog Post], 2018. URL <https://www.topbots.com/most-important-ai-research-papers-2018/>. Retrieved from <https://www.topbots.com/most-important-ai-research-papers-2018/>.
- Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., and Lipson, H. Understanding neural networks through deep visualization. *Deep Learning Workshop, International Conference on Machine Learning (ICML)*, 2015.