

Improving Model-Based Reinforcement Learning by Disentangling the Latent Representation of the Environment

Abstract

This thesis explores to what degree model-based reinforcement learning can benefit from recent advances in representation learning. Specifically we measure the impact that the amount of feature-entanglement within the learned representation of the environment influences overall model performance. We train a total of 45 (variational) autoencoders on a custom box-physics environment, varying the relative influence of the Kullback-Leibler divergence term on the encoders loss. For each of these models, we measure the amount of feature entanglement in their latent representations using the measures proposed in [Higgins et al. \(2017\)](#) and [Eastwood, C., & Williams, C. K. \(2018\)](#). These disentanglement scores will then be evaluated against the loss of a recurrent LSTM network that was pre-trained on sequences of environment encodings, generated by the relevant autoencoder. -- We find that less entangled representations of the environment significantly increase the accuracy of the recurrent model and that this effect is even stronger for larger latent spaces.

1. Introduction

This research is focused on improving the performance of model-based reinforcement learning. Specifically we attempt to increase a networks ability to to predict its environment. We do not consider to what extent this increases performance of the agent as a whole.

Model-based reinforcement learning is centered around the distinction between understanding the world and acting within it. This was found to be useful, since model-free agent architectures struggle with the credit assignment problem and are therefore limited to rather small network sizes^[Source 1]. In the context of reinforcement learning, credit assignment means figuring out specifically which past action that the agent performed is responsible for receiving a future reward.

Model-based methods attempt to solve this issue by training separate networks for understanding the world and acting within it. The acting part of the agent will further be referred to as the 'controller'. It is trained to solve the underlying problem and is tasked with learning how to maximize reward. It does that by considering the information-rich output of the first network, which is learning a representation of the environment. The representation network will further be referred to as the 'world-model'.

The distinction of an agent into controller and world-model has been found by [Atkeson, C. G., & Santamaria, J. C. \(1997\)](#) to require significantly fewer data and even exceeding the performance of model-free methods with better trajectories, plans and policies . This is especially useful in the field of Robotics where, given that no perfect simulation of the environment is available, which is usually the case, acquiring data is often very expensive. Improvements on the performance of model-based reinforcement learning are therefore potentially very useful.

As mentioned earlier this work does not consider the performance of the whole agent. Specifically this means that the controller is out of focus while we try to improve the prediction-accuracy of the world-model.

As done in [Ha, D., & Schmidhuber, J. \(2018\)](#), we separate said world-model into an autoencoder network that extracts features of the environment and a recurrent network that predicts the dynamics of the world - how it changes over time.

In this thesis, we will explore how different architectures for feature extraction can improve the predictive performance of the world-model. We will specifically focus on the aspect of disentanglement of the extracted features.

Disentanglement of features means keeping different features separated from each other i.e. a value that has been extracted should only be responsible for one distinct feature of the environment. This could for example mean that we have a value which encodes only the hair-color of a person rather than, say also be partially responsible for the shape of her eyes. This has been argued to be likely useful by [Bengio, Y., Courville, A., & Vincent, P. \(2013\)](#). In addition to the potential increase in model performance, disentanglement provides benefits for the field of explainable artificial intelligence as well.

We aim to answer the following research question:

To what extent does the disentanglement of features in the input improve the performance of a recurrent neural network?

Answering this question contributes to the field of representation learning, which has gained popularity in and around 2013 [Bengio, Y., Courville, A., & Vincent, P. \(2013\)](#). [Bengio et al.](#) argues that an automatic learning of disentangled features leads to a faster development cycle of novel applications and is an important step toward [general] artificial intelligence.

"Learning an interpretable factorised representation of the independent data generative factors of the world without supervision is an important precursor for the development of artificial intelligence that is able to learn and reason in the same way that humans do." - [Higgins et al. \(2017\)](#)

The feature extractions will be performed by an autoencoder. Following we will introduce autoencoders and then their improved variational and β -variational variants.

Autoencoder

Autoencoders (AE's) are an unsupervised learning technique, meaning that, as opposed to predicting labels of the data they attempt to extract meaningful features of it. The performance of an autoencoder is measured by its ability to recreate some original input. This is difficult because Autoencoders pass the input through a smaller, hidden layer of lower dimensionality called the 'latent space representation' or 'latent vector'. This vector's values are said lay somewhere in 'latent space'.

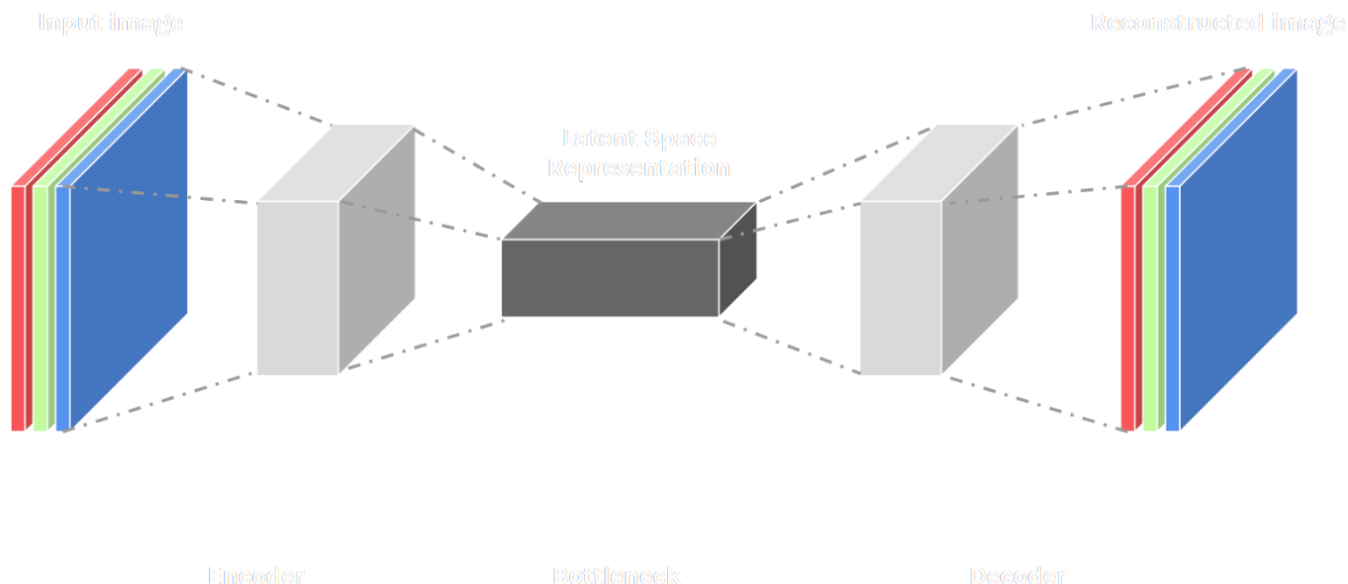


Figure 1: The left part of the network, mapping the input data onto the latent space is called the 'encoder' while the rest of the network can be referred to as either the 'decoder' or 'generator'. The latter recreates an image based on the latent representation generated by the encoder.

This network is trained to minimize the difference between its input and output. Common difference functions that are used to quantify this are the '*mean square error*' and '*binary cross-entropy loss*', the choice of either depending on the format of the input. The loss function for training an autoencoder uses one of these functions;

$$\text{loss} = \text{dif}(\text{original}, \text{reconstructed})$$

In theory the encoder is forced to learn some meaningful representation of features in the data. This is because the decoder has to rely on this information to reconstruct the input as well as possible. Providing a meaningless or incomplete encoding to the decoder will worsen its ability to do so. In practice, vanilla autoencoders tend to map certain classes of the input onto some range in euclidean space e.g. in the context of the MNIST digit dataset it may represent the twos as an activation in the range of $[1, 3]$ and the fours at $[5, 8]$. While this leads to high reconstruction performance, the encoder is no longer incentivized to learn 'what makes a two' i.e. its features.

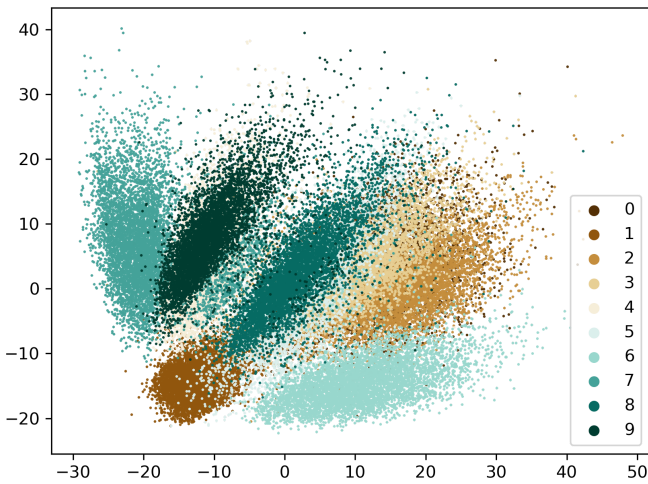
Variational Autoencoder

Variational autoencoders (VAE's) are incentivized to have their latent space representation follow the standard normal distribution $N(0, 1)$. This means that, for all training inputs to the encoder, any factor of the latent vector should have a mean that is close to zero and a standard deviation close to one. This forces the encoder to generate meaningful representations it has to fit the encoding into a smaller range inside the latent space. An added benefit to this compactness is that traversing the latent space in order to generate novel reconstructions leads to more meaningful results [Spinner et al., \(2018\)](#).

Regular Autoencoder

Variational Autoencoder

Regular Autoencoder



Variational Autoencoder

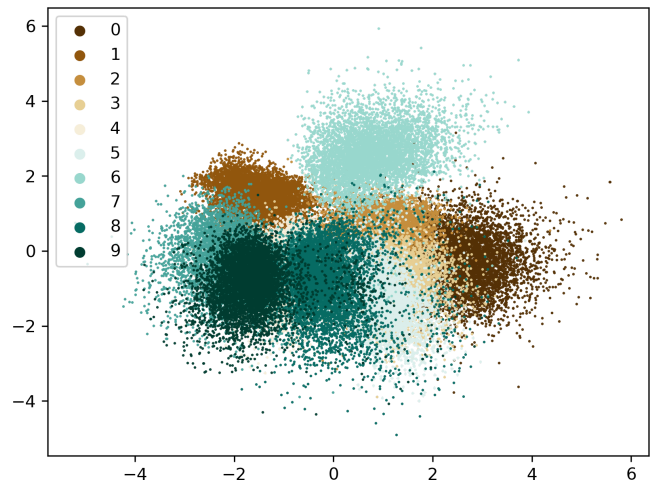


Figure 2: A visual comparison between the latent space of a variational non-variational autoencoder.

The incentive to generate a normally distributed latent space representation is given by a second optimization target during training of the neural network. Specifically, the encoder should generate an output that is similar to that of the standard normal distribution $N(0, 1)$. The first step in implementing this is to separate the output of the encoder into the mean μ and standard deviation σ^2 of a normal distribution $N(\mu, \sigma^2)$. During training, the actual latent representation is then sampled from this distribution. When more accuracy is required while deploying the network μ can directly be used as the latent vector. In addition to splitting the encoders output another term is added to the loss function of the network. The Kullback-Leibler distance can be used for that, since it is a measure of how similar one probability distribution is to another. The loss function now reads; $\text{loss} = \text{dif}(\text{original}, \text{reconstructed}) + \text{kl_divergence}(\text{latent_space}, N(0,1))$

This new structure leads to more disentanglement but comes at the cost of some reconstruction performance. This is natural since a limit on the range of latent activations practically reduces the information capacity of the bottleneck channel. On the example of MNIST, [Spinner et al., \(2018\)](#) observed that in order to reach the performance of an AE with two latent variables a VAE already requires at least four latent variables.

β Variational Autoencoder

β variational autoencoders (β -vae's) provide a simple but useful addition to regular VAE's in that they hyper-parameterize how important it is for the latent space representation to follow the normal distribution $N(0,1)$. This means adapting the loss function to include the β term as a factor on the Kullback-Leibler penalty.

$$\text{loss} = \text{dif}(\text{original}, \text{reconstructed}) + \beta * \text{kl_divergence}(\text{latent_space}, N(0,1))$$

[Higgins et al. \(2017\)](#) found that;

"When compared to the unmodified VAE baseline ($\beta = 1$), β -VAE consistently learns significantly more disentangled latent representations." - [Higgins et al. \(2017\)](#)

They also found that β -vae's are able to produce latent representations with an amount of disentangling comparable to contemporary state of the art architectures, like InfoGAN and DC-IGN.

Considering the drawbacks of regular variational autoencoders ($\beta = 1$) it is only natural to assume that an increased β further reduces the reconstruction performance of the autoencoder. This is indeed what [Higgins et al. \(2017\)](#) found when visually inspecting their results.

To mitigate this, [Burgess et al. \(2018\)](#) proposes 'controller capacity increase', a training technique that diminishes this effect. As the name suggests, this means starting of with a high β and low information capacity and further reducing β during training to progressively allow more features to be expressed. This makes the encoder learn the most important features (e.g. position) first and only later also capture details (e.g. rotation) while keeping the important features disentangled. [Burgess et al. \(2018\)](#) found this technique to be very effective, leading to a lower reconstruction loss and visually impressive results.

Materials and Methods > How did you do what you did?

Environment

The autoencoders were trained on an open-ai gym environment that we created for this research. It features some simple box-physics with objects falling from the top of the screen and colliding with each other on the ground. We additionally included a static triangle on the bottom for some more interesting interactions.



We decided that we would optimally try to extract features from an environment that is rich in pose-information, with interesting time-dynamics and a real-world relevance. Since the focus of this work lies on modeling the world rather than acting within it we also preferred an environment without actions. We chose to create our own environment because none of the other existing open-ai gym environments tick all of these boxes, arguably because the whole framework is very action-centric.

The observations we sampled from the environment were 64 x 64 pixels large with a single gray-scale channel. In addition, the environment provides its instantiation parameters for analysis. Those are the x, y position $[0, 1]$ and $\sin(\alpha), \cos(\alpha)$ for each dynamic object in the environment (α being the rotation of the object in radians) as well as the x position of the triangle.

β -Variational Autoencoders

We used the β -variational architecture that was proposed by [Burgess et al. \(2018\)](#), which is a slightly adapted version of the structure originally proposed in [Higgins et al. \(2017\)](#).



Figure 3: Architecture of the β -variational autoencoder, as proposed by Burgess et al. (2018). Encoder: all ReLU, 4 convolutional layers (32 channels, 4x4 kernels, stride of 2), 3 fully connected layers (2x size of 256 and 1x size of 2 * latent space for mean and std). Decoder: Inverse of encoder. The bottleneck is twice the size of the latent space because it contains both the mean and standard deviation of a normal distribution. We split it in half when using the model.

We trained a total of 45 β -vae's for $2 \times 100,000$ iterations on mini-batches consisting of 50 images of randomly instantiated environments. The 45 models break down like this;

Latent Space Sizes	Betas	+ 1
\$6, 12, 24, 64\$	\$0, 0.2, 0.4, 0.6, 0.8, 1, 1.2, 1.4, 1.6, 1.8, 2\$	decoder trained on instantiation parameters

The optimizer used is Adam with a learning rate of 0.002 for the first and 0.003 for the second half of training. If one of the autoencoders performs significantly worse than average we retrain the network up to 3 times, decreasing both learning rates by 0.0005 each time.

Training the autoencoders on our environment, they often learned to ignore all the dynamic objects and tended to only predict the triangle with a solid color background instead. This was even more of a problem for higher amounts of β . To counteract these issues we initially decreased the loss for wrongly predicting pixels that display the background in the input image. Conceptually this gives the network more room to try and predict objects without immediately leading to a large increase of the loss. We found that initial values of about 0.5x the relative amount of object-pixels vs background-pixels worked very well. During the training process after the first 100,000 iterations we double that factor and linearly increase it to 1 for the last 100,000 iterations.

Recurrent LSTM Predictor

The recurrent neural network we used as the performance metric is based on Ha, D., & Schmidhuber, J. (2018). We use a Long-Short-Term-Memory network with 256 hidden units. Given the latent space encoding of one of the encoders it has to predict the latent vector of the next state of the environment.

Instead of a 'mixture-density' network as suggested in Ha, D., & Schmidhuber, J. (2018), we use a regular LSTM model. Mixture-density models are used to express an amount of uncertainty over the predictions of the network. Since the environment we use of deterministic nature, uncertainty is not as useful. Regardless, we do not aim to optimize the LSTM's performance anyway but use it as a metric instead.

For each of the β -Variational autoencoder we pre-train the predictor for 5000 iterations of 50 frames each so that it can learn the basic latent dynamics of the environment. After training we evaluated the model for another 100 iterations. We then used the average loss on these last 100 iterations as the inverse prediction-score.

Disentanglement Metric

In order to relate the performance of the recurrent neural network to the amount of disentanglement in the latent space we need a way to measure the latter. While it is possible to visually evaluate the amount of disentanglement, a quantitative measure is optimal.

We considered two different metrics, most prominently the methods proposed in [Higgins et al. \(2017\)](#) and [Eastwood, C., & Williams, C. K. \(2018\)](#).

Following we lay out the procedure suggested by [Higgins et al. \(2017\)](#) in the original paper on β -vae's;

1. Choose one factor k of the latent space
2. Iterate:
 1. Sample two latent vectors v_1, v_2 for which $v_1[k] = v_2[k]$
 2. Decode v_1, v_2 as p_1, p_2
 3. Re-encode p_1, p_2 as e_1, e_2
 4. Calculate $\text{difference}(e_1, e_2)$
3. Take the average of $\text{difference}(e_1, e_2)$ as avg
4. How accurately can you predict k from avg ?

This method is very convenient because it doesn't need a ground truth to evaluate the disentanglement. However, it seems that the result of this procedure will depend to a large extent on the reconstruction performance of the decoder.

Since we have the environments instantiation parameters readily available as a ground truth, we chose to implement the metric proposed by [Eastwood, C., & Williams, C. K. \(2018\)](#) as well. To understand this metric, entanglement can be thought of as a measure of how many features a single latent factor k predicts. Optimally, each k is responsible for only one feature f , which would be a completely disentangled representation. However, a representation in which f is encoded by both k and another factor l is still fully disentangled as long as k and l only predict f . This would be a less efficient encoding though.

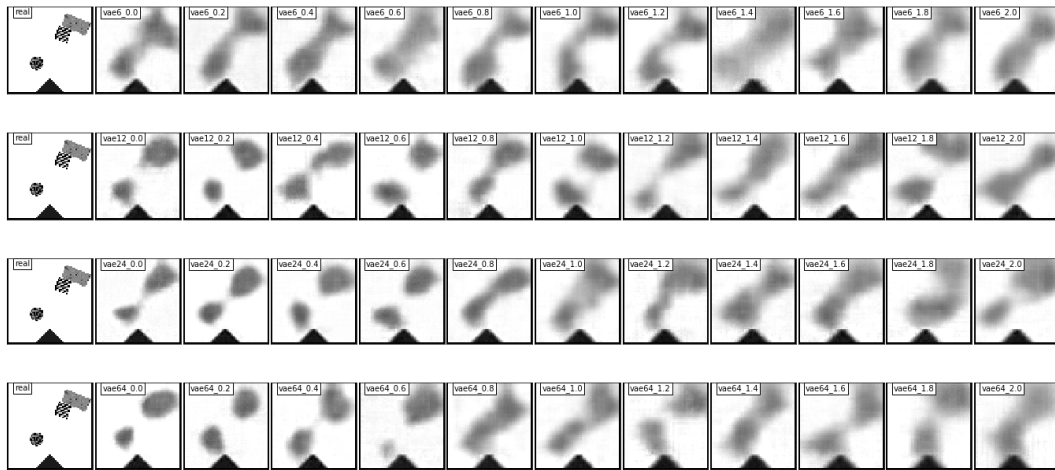
To analyze this relationship between factors and features, [Eastwood, C., & Williams, C. K. \(2018\)](#) suggest,

1. let g be the ground truth of features
2. for each factor k in the latent space:
 1. fit linear regressor (Lasso) to predict k given g
 2. add resulting coefficient vector as a row of the coefficient matrix C
3. transform and analyze C for a disentanglement and completeness
4. use regressor loss as a measure of informativeness

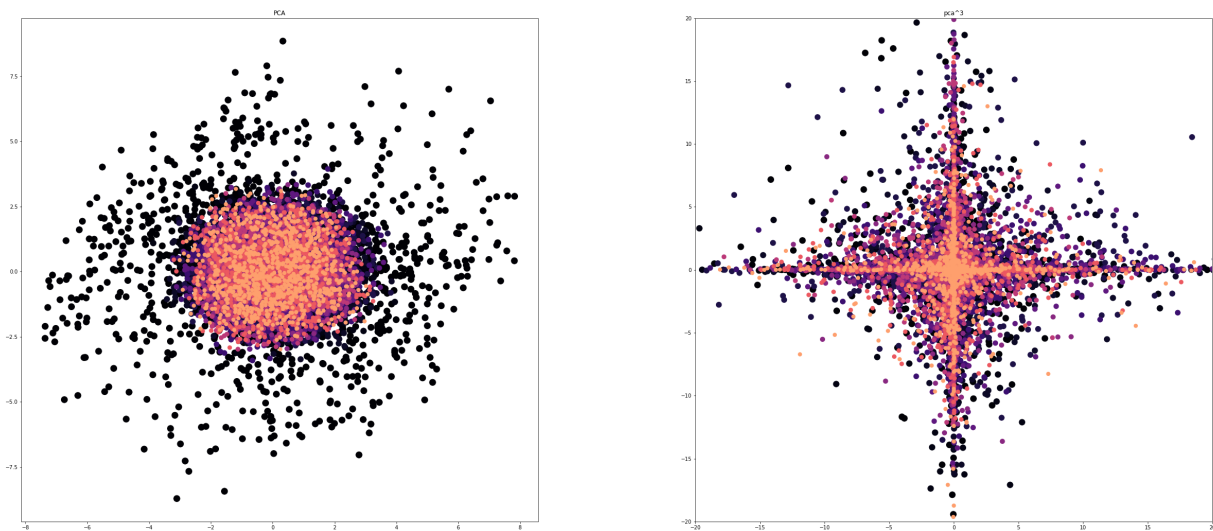
Since we implemented both of those measures we will

Results

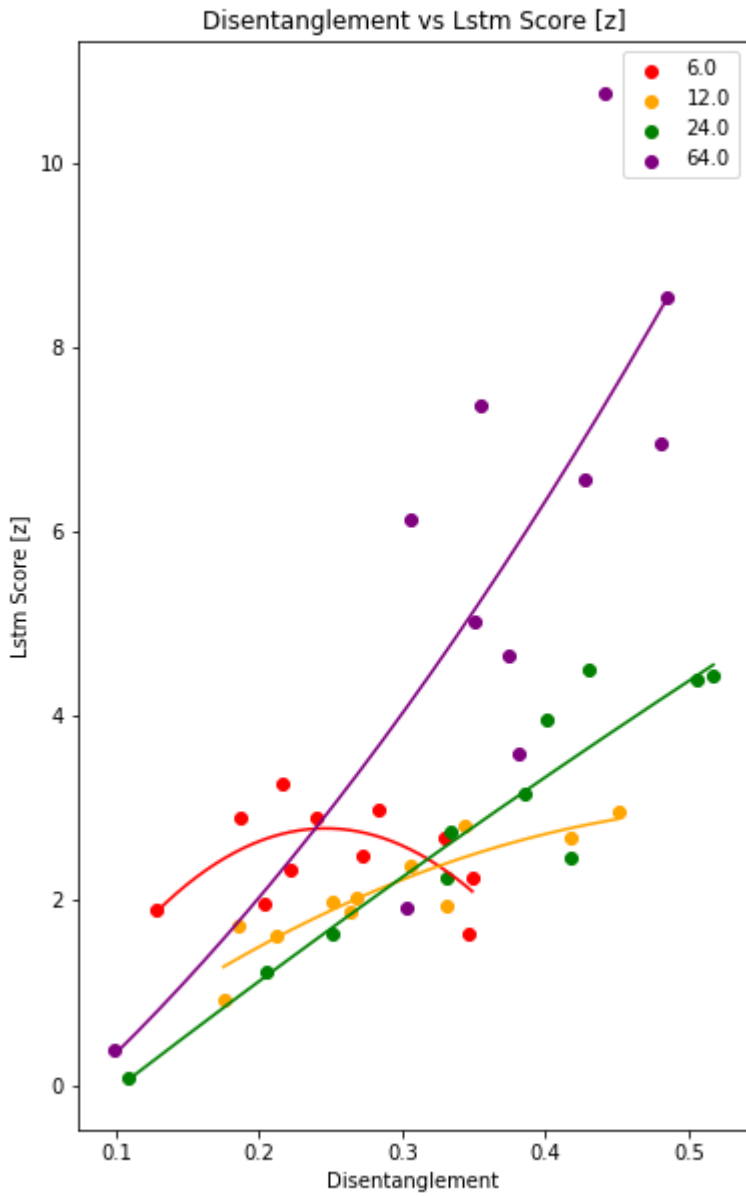
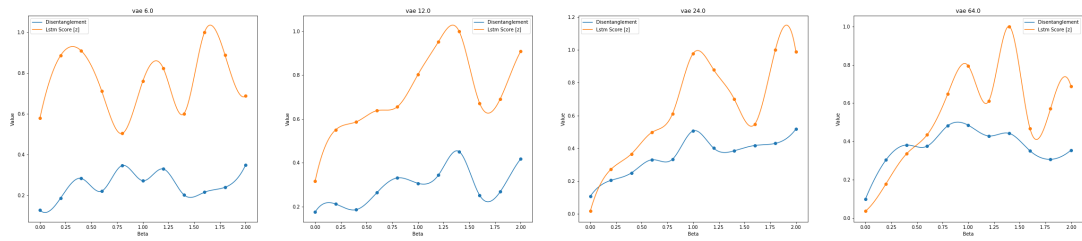
Autoencodings



Distribution



Disentanglement Correlations



Significance Testing

Discussion

Conclusions

It works!

References

- [1]: Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., ... & Lerchner, A. (2017). beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations* (Vol. 3).
- [2]: Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., & Lerchner, A. (2018). Understanding disentangling in β -VAE. *arXiv preprint arXiv:1804.03599*.
- [3]: Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8), 1798-1828.
- [4]: Spinner, T., Körner, J., Görtler, J., & Deussen, O. (2018). Towards an Interpretable Latent Space: an Intuitive Comparison of Autoencoders with Variational Autoencoders. In *IEEE VIS 2018*.
- [5]: Ha, D., & Schmidhuber, J. (2018). Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems* (pp. 2450-2462).
- [6]: Atkeson, C. G., & Santamaria, J. C. (1997, April). A comparison of direct and model-based reinforcement learning. In *Proceedings of International Conference on Robotics and Automation* (Vol. 4, pp. 3557-3564). IEEE.
- [7]: Hinton, G. E., Krizhevsky, A., & Wang, S. D. (2011, June). Transforming auto-encoders. In *International Conference on Artificial Neural Networks* (pp. 44-51). Springer, Berlin, Heidelberg.
- [8]: Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.
- [9]: Eastwood, C., & Williams, C. K. (2018). A framework for the quantitative evaluation of disentangled representations.