Research project: Gestalt code analysis, regularisation and closed-loop extension for a self-supervised location and identity tracking system

Simon Frank

Submission on March 14, 2023
WS 22/23

Prof. Dr. Martin V. Butz M.Sc. Manuel Traub

Dept. of Computer Science
Neuro-Cognitive Modeling
University Tuebingen

Abstract

Recent object reasoning datasets, such as CATER, have revealed fundamental shortcomings of current vision-based AI systems, particularly when it comes to explicit object representation, object persistence, and object reasoning. A self-supervised LOCation and Identity tracking system (Loci) has been introduced that excels at the CATER tracking challenge. Loci tackles the binding problem by processing separate, slot-wise encodings of 'what', the Gestalt code, and 'where', the position code. During the project, the Gestalt codes were analysed. It was shown that the binary Gestalt codes are disentangled, which may be a reason for the superior performance of Loci. Various regularisation techniques were also tested during the project, but none could outperform the binary version. The architecture of Loci was extended to include a prediction module in order to be able to run closed-loop with only the Gestalt and position codes. Through special training, the prediction was correct 20 time steps into the future using the CLEVRER dataset. This method provides a computationally fast and accurate way of reasoning at conceptual levels.

Contents

List of Figures

List of Tables

1 Introduction

Although neural networks have achieved good results in the vision domain, state-of-the-art video reasoning systems still struggle to model fundamental physical properties of objects. This shortcoming can be overcome by solving the binding problem in artificial neural networks. The binding problem describes the fundamental challenge of how to disentangle and uniquely encode entities while flexibly binding the what to the where entity. A sub-problem of the binding problem is the separation problem. It describes the challenge of identifying different objects, their properties and locations within the perceived scene, and disentangling them into separate latent entity property and location codes. Inspired by the theory of what and where processing via ventral and dorsal signal processing pathways, respectively, a **Loc**ation and **i**dentity tracking system (Loci) has been introduced. Loci tackles the binding problem through a coding mechanism that splits the latent state of an object into a position and a gestalt-based coding. Loci has already outperformed current state-of-the-art architectures in the CATER-tracking challenge. Since the Gestalt codes played a major role in the success of both challenges, they will be the subject of this research project.
The first step is to analyse and compare the different Gestalt codes produced and used during the different tasks. Therefore, the high dimensional Gestalt Codes will first be reduced to two and then clustered. This procedure will provide insight into the different versions of Gestalt codes. The disentanglement of the Gestalt codes will also be analysed. In the next step, different regularisation methods are applied to the Gestalt codes and their effects are analysed. Finally, Loci will be extended to run in a closed-loop using only the Gestalt codes and not the generated images. This extended version of Loci will be specially trained to run in a closed-loop.

2 Methods and Results

2.1 Data analysis of the Gestalt codes

The first step in the project is to cluster the Gestalt codes generated by Loci to gain a better understanding of their structure. For this purpose, a data analysis pipeline was implemented and used throughout the project. The architecture of this pipeline is described in the following section. This pipeline will then be applied to the Gestalt codes.

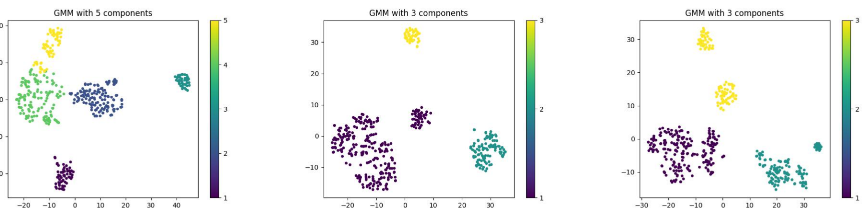
2.1.1 Data analysis pipeline

For the later analysis, 450 Gestalt codes generated by the transition model were stored, as this number provides enough variation in the data for analysis, but not too many data points to slow down the process and make the analysis unfeasible. In order to cover a wider range of different objects, only the Gestalt codes produced at time $t = 0$ were used. This allowed 45 different videos to be used as input, as Loci is configured to expect ten individual objects.
The Gestalt codes are high-dimensional data (24, 48 or 96 dimensions, depending on the configuration). Since reducing the dimensionality of data can improve the efficiency and effectiveness of clustering algorithms [2], the dimensionality of the stored Gestalt codes will be reduced using t-SNE. t-SNE (t-Distributed Stochastic Neighbour Embedding) is a popular dimensionality reduction method that is particularly well suited for visualizing high-dimensional data in two or three dimensions [13]. It works by constructing a probability distribution over pairs of high-dimensional data points and then minimising the Kullback-Leibler divergence between the distribution and a lower-dimensional representation of the data. This allows t-SNE to preserve the local structure of the data while reducing the dimensionality of the data.
In this project *openTSNE* was used to project the high dimensional Gestalt codes to two dimensions. *openTSNE* is a modular Python implementation of t-Distributed Stochastic Neighbour Embedding (t-SNE) that incorporates the latest improvements to the t-SNE algorithm, including the ability to add new data points to existing embeddings and massive speed improvements [8].
The next step in the pipeline is to apply t-SNE to the stored high-dimensional Gestalt codes to reduce the dimensionality to two. These two-dimensional Gestalt codes are then clustered. The clustering algorithm used is the Gaussian Mixture Model (GMM). The Gaussian Mixture Model is a popular method for clustering data, which assumes that the data is generated from a mixture of several underlying Gaussian distributions [9]. One of the advantages of the GMM is that it can handle data that is not necessarily linearly separable, as it allows for the creation of more complex cluster shapes. In addition, the GMM can handle data with multiple modes, or regions of high

density, making it well suited to data that may not have a clear separation between clusters. To fit a GMM to a data set, an expectation maximisation (EM) algorithm can be used to estimate the parameters of the Gaussian distributions. To cluster the Gestalt codes, the *GaussianMixture* of *sklearn* [7] was used in this research project.
A crucial point in clustering is choosing the right number of clusters since the number of clusters has to be predefined by the user. To solve this problem, the silhouette score was used. The silhouette score is a popular measure to evaluate the performance of a clustering algorithm. It calculates the average silhouette value of all samples in a cluster, where the silhouette value is a measure of how similar a sample is to its own cluster compared to other clusters. Higher silhouette values indicate a more cohesive cluster, and lower silhouette values indicate a cluster that may be too dispersed or overlapping with other clusters. Therefore, the silhouette score can be used as a guide for choosing the optimal number of clusters, as a higher silhouette score usually indicates that the chosen number of clusters is appropriate for the data [10].
To determine the silhouette score for a set of 450 Gestalt codes, the GMM algorithm is run 5 times in a row with a fixed number of predefined clusters, varying the number of clusters between two and 20. The average silhouette score is then calculated for each number of clusters. The number of clusters with the highest silhouette score is then used to cluster the Gestalt codes for later analysis.

2.1.2 Clustering

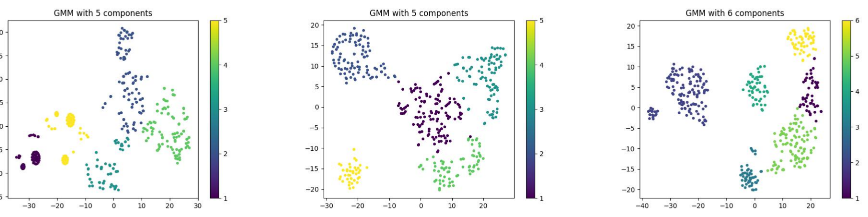
The pipeline described in 2.1.1 was applied to the Gestalt codes generated by Loci using the CATER dataset and an aquarium video found on YouTube, because these two datasets are completely different and one would expect a significant difference in the clustering of the Gestalt codes. In the CATER dataset, various geometric objects rotate, move and jump in a scripted random way [5].
For the CATER dataset, the size of the Gestalt code was varied between 24, 48 and 96 bits, and the Gestalt codes before and after the transition module were also analysed. While this analysis was all done with a binary Gestalt code, as used in [12], 24 bit floating point Gestalt codes were also analysed. This allows the effects of different sizes and regularisation on the Gestalt code to be studied. For the aquarium footage, a 192-bit binary Gestalt code was used, as more information and bits are required to model the large variation of complex fish. There is no difference in the separation and spread of the different clusters between the 24, 48 and 96 bit binary Gestalt codes (Figure 1). In general, most of the clusters are tightly packed, so there is little intra-class variance in these clusters, but there is at least one cluster that is more widely spread. The images generated using these



(a) 24 bit Gestalt codes (b) 48 bit Gestalt codes (c) 96 bit Gestalt codes

Figure 1: Gaussian Mixture Model clusters of the binary Gestalt codes, extracted before the transition module, after dimensionality reduction with t-SNE using the CATER dataset

Gestalt codes as input to the decoder module do not yield any meaningful geometric objects. This observation makes sense because the decoder is only trained with the Gestalt codes generated by the transition module, and not those generated by the encoder module, which are input to the transition module.



(a) 24 bit Gestalt codes (b) 48 bit Gestalt codes (c) 96 bit Gestalt codes

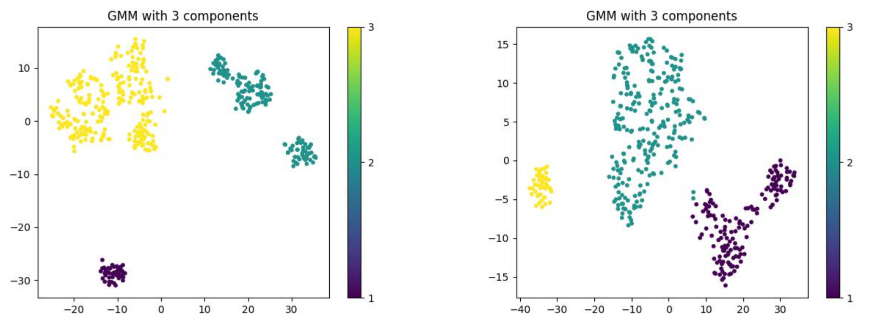
Figure 2: Gaussian Mixture Model clusters of the binary Gestalt codes, extracted after the transition module, after dimensionality reduction with t-SNE using the CATER dataset

The data points for all three different dimensions of the Gestalt codes extracted after the transition module and binarization are more spread out compared to those extracted before the transition module (Figure 2). Only the clusters of the 96-bit Gestalt code have slightly less intra-class variance and slightly more inter-class variance. In general, the amount of clusters is higher compared to the number of clusters when the Gestalt codes were extracted before the transition module. It can be said that more variability is needed to generate the images of the different objects by the decoder module using the binary Gestalt codes.

For further analysis, the Gestalt codes from the different clusters were used to generate images by the decoder model. The contents of these clusters, separated by the dimension of the Gestalt code, are listed below:

1. 24 bit
    (a) Various coloured cones, cubes and balls with metallic surface
    (b) Empty and pink balls
    (c) Multicoloured cones, balls and cubes with a matt surface
    (d) Empty
    (e) Assorted blue balls and cubes with a matt surface
2. 48 bit
    (a) Multicoloured cones with a matt surface
    (b) Empty
    (c) Mixed coloured balls and cones with a metallic surface
    (d) Multicoloured cubes with a matt and metallic surface
    (e) Small blue spheres with a matt surface
3. 96 bit
    (a) Multi-coloured cubes and spheres with a metallic surface
    (b) empty
    (c) Small blue balls with a matt surface
    (d) Mixed coloured balls and cubes
    (e) Multicoloured cones, balls and cubes with a matt surface
    (f) Multicoloured cones with a metallic surface

The empty classes contain Gestalt codes that do not generate an image. This can be explained by the fact that the Loci architecture is configured to expect ten different objects to be tracked over time with a single Gestalt code. Since there are not always ten objects in the input images, some Gestalt codes must produce an empty image. In all three cases there are classes, e.g. small pink and blue balls, which can be seen as artefacts.
When a class contains different objects, there is also a variation in the colour of the objects. It can be assumed that colour is a feature that does not have a large impact on the Gestalt code, so it may only take a few bits to change the colour of an object. In most classes there are also different objects, so the same can be said for the object encoding bits. But there are no classes containing objects with matt and metallic surfaces. It seems that most of the bits are needed to encode the surface and therefore the lighting of an object.
Even though in the final version of Loci the Gestalt codes are always binary, it is still interesting to analyse the clustering of the Gestalt codes when they are generated by a variational encoder, where no regularisation is applied to them and they are 24 dimensional.



(a) Gestalt codes extracted before the transition module (b) Gestalt codes extracted after the transition module

Figure 3: Gaussian Mixture Model clusters of the floating point 24 dimensional Gestalt codes, extracted before and after the transition module, after dimensionality reduction with t-SNE using the CATER dataset

In contrast to the binary Gestalt code clusters, the floating-point Gestalt codes can be clustered with only 3 clusters, which are well separated, this is true for the Gestalt code extracted before and after the transition module (Figure 3). The three classes contain the following objects:

1. empty
2. Various coloured cubes, cones and spheres with metallic and matt surfaces
3. Small blue balls with a matt surface

With the continuous Gestalt codes, there is now one large class containing all meaningful geometric objects, while in the binary case there are many classes for the different objects and surfaces. The assumption can be made that there is no need to disentangle the different features such as shape, colour and surface into separate subspaces if no additional regularisation, such as binarization, is applied to the Gestalt codes.

The five clusters of Gestalt codes from the aquarium footage are more spread out and not clearly separated from each other (Figure 4).
It can be concluded that for complex figures with high intra-class variance, such as fish from the aquarium footage, the whole high-dimensional space of the Gestalt code must be used. The data points do not lie on a data manifold as for simple geometric objects from the CATER dataset.
Through this analysis, meaningful clusters in the binary Gestalt code have been identified, providing a basis for further investigation.

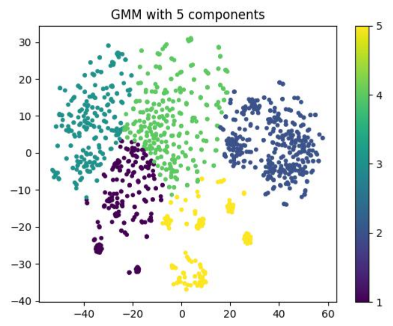


Figure 4: Gaussian Mixture Model clusters of 192 bit binary Gestalt codes, extracted after the transition module, after dimensionality reduction with t-SNE using aquarium footage

2.1.3 Inspection of the principal axes

The content of the different classes of Gestalt code suggested that there are subspaces in the high-dimensional Gestalt code that independently change the colour, shape and surface of the geometric objects. If this were the case, Loci would be able to produce disentangled Gestalt code through the architecture, which could be one reason for the good results in the CATER-tracking challenge.
A good tool for finding these disentangled subspaces or axes in the Gestalt code is Principal Component Analysis (PCA). Principal component analysis is a technique used primarily to reduce the dimensionality of a data set while preserving as much of the variability in the data as possible. PCA finds a new set of variables, called principal components, which are linear combinations of the original variables and capture the most important patterns in the data. PCA works by identifying the directions in the data that have the largest variances, which are then used to construct the principal components. These principal axes are ranked by the amount of variance they explain [1].
As Loci achieved the best results in the CATER-tracking challenge with the binary 96-bit Gestalt code, only this type of Gestalt code is used for further analysis. The effect of the main axes found on the colour, shape and surface of the objects was investigated using the procedure described below, to make statements about the disentanglement.
First, a fixed set of Gestalt codes was selected. This could be the full set of 450 Gestalt codes or just a subset using the classes found in 2.1.2. PCA is then used to determine the main axes of variance in the selected set of Gestalt codes. A random Gestalt code from the subset was selected and modified by subtracting or adding a portion of the axes, or in other words, the Gestalt code was used as a starting point and then new Gestalt codes were sampled moving down and up the selected axes. Since the starting point (chosen

Gestalt code) is always different, the step size in both directions must be adjusted accordingly to get a meaningful result. These new Gestalt codes are no longer binary, but were still clipped at 0 and 1 to create meaningful inputs to the decoder module, which was used to generate the corresponding image for each sampled Gestalt code.

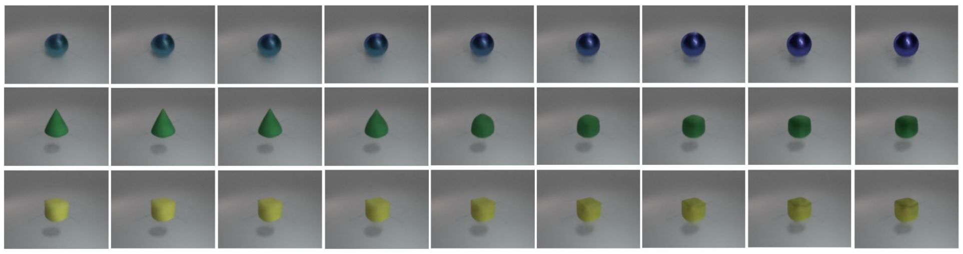


Figure 5: Sampling of new objects along the main axis of variance across all 450 Gestalt codes and three different starting objects

Figure 5 shows the effect of the first-principle axes on three different objects when all 450 Gestalt codes are used to compute these axes. You could say that the first-principle axis shows what most of the bits are used for and what they encode. For all three objects there is no change in colour, but a change in surface. The metallic appearance of the surface is increased. In the case of the cone, the object itself changes to a cube. This observation supports the assumption that colour is not a dominant feature in the latent Gestalt code space, while the surface is the most dominant feature. In the

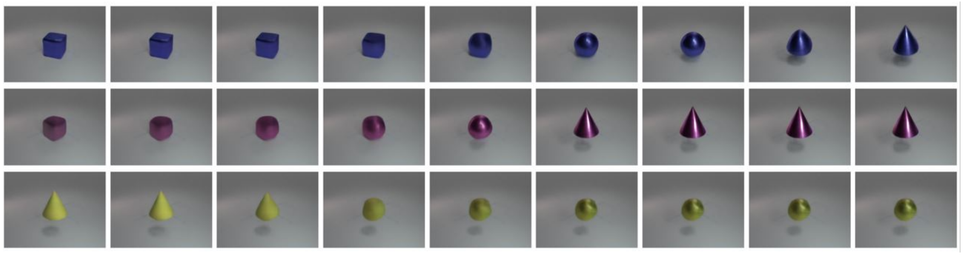


Figure 6: Generating new objects using the found shape axis

first class defined in 2.1.2 only the shape of the different objects changes. This suggests that the main axis of variance in this class is responsible for changing the shape of an object. Using these axis to generate new shape codes and corresponding images, only the shape changes while the colour remains the same for three different starting objects with matt and metallic surfaces. For a starting object with a matt surface, the surface of the object also becomes metallic (Figure 6). The found form axis is not a pure form axis depending on the start form code. When the first principle axis of class five is used, only the colour changes, while the surface and shape remain the

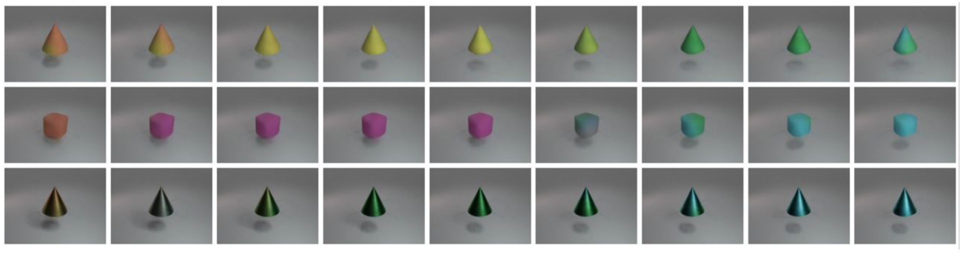


Figure 7: Generating new objects using the found colour axis

same for matt and metallic objects of different shapes (Figure 7). This axis is therefore a pure colour axis.
If the newly found colour and form axes were to be disentangled, it would be possible to use them in an additive manner, changing colour and form simultaneously and independently of each other, as shown in Figure 8. In

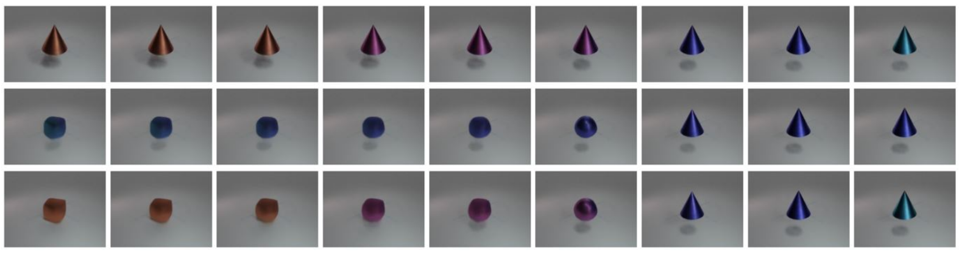


Figure 8: Generate new objects using the colour axis (**top row**), the form axis (**middle row**), both axes in an additive manner (**bottom row**)

the first row, the colour axis is applied, in the second the shape axis is applied and in the third both axes are used in an additive manner. Since the resulting objects in the third row have the colour of the first row and the shape of the second row, it can be concluded that the colour and shape axes are disentangled and only change this specific feature of an object. This confirms the assumption that Loci produces disentangled Gestalt codes.
Data analysis of the Gestalt codes showed that the high-dimensional Gestalt codes lie on a data manifold when binary regularisation is applied. This also ensures that there are disentangled form and colour axes. These two attributes, which are most dominant in the binary Gestalt code, play a crucial role in the good performance of Loci in the CATER-tracking challenge.

2.2 Regularization of the Gestalt codes

During the development of Loci, binarising the Gestalt codes after the transition module and before the slot-wise encoder gave surprisingly good results, especially on the CATER-tracking challenge, although the amount of information a Gestalt code can represent was dramatically reduced by this strong

inductive bias. The reasons for the good performance may be the results found in 2.1.2 and 2.1.3, the distinct clusters and the disentangled Gestalt codes. This raises the question of whether there would also be distinct clusters and disentangled axes in the Gestalt codes if a different regularisation were applied. This will be investigated in the following section.
First, the regularisation is changed so that the resulting Gestalt codes are no longer binary, then the Gestalt code is changed into a categorical distribution by another regularisation technique. For all regularisation techniques, Loci was trained on the CATER dataset in a similar way as in [12], first for 24 hours with a higher learning rate and lower resolution, followed by 8 hours of training with a lower learning rate but at full resolution. This setup provided a good trade-off between minimal training time and meaningful results. The same training procedure was used throughout the project in order to obtain comparable results. All Gestalt codes were extracted after the transition module and processed with the same pipeline described in 2.1.1.

2.2.1 Non binary Gestalt codes

The binarization consists of two steps: First, the sigmoid function is applied to the Gestalt code G, and then some random noise is added in the following way [12]:

$$G = G + G(1 - G)\mathcal{N}(0, 1) \tag{1}$$

Two different settings have been tested. In the first setting, both binarisation steps were omitted to investigate the effect of a completely unconstrained Gestalt code. In the second setting, the sigmoid function is not applied, but noise is added as in equation 2.2.1. Figure 9 shows the clus-

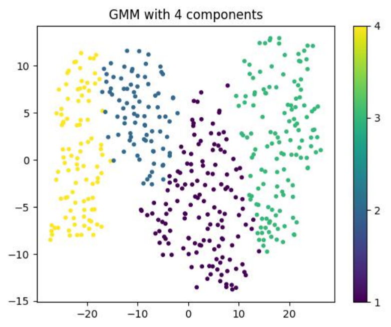


Figure 9: Gaussian Mixture Model clusters of unconstrained Gestalt codes, extracted after the predictor, after dimensionality reduction with t-SNE using the CATER dataset

tered Gestalt codes when no regularisation is applied. The clusters are not well separated and there is a large intra-class variance. The resulting images and clusters are also not meaningful. The hypothesis can be formulated that without regularisation, the Loci (at least with the training procedure used) cannot find a good solution in the high dimensional Gestalt code space. With the second regularisation setting, the clusters are slightly more sepa-

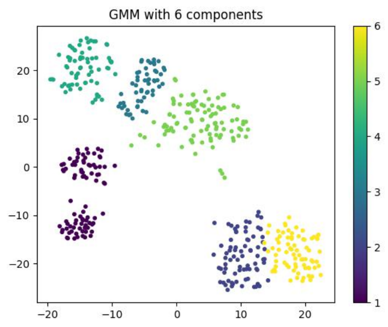


Figure 10: Gaussian Mixture Model clusters of Gestalt codes constrained with random noise, extracted after the predictor, after dimensionality reduction with t-SNE using the CATER dataset

rated and there is less intra-class variance (Figure 10). The resulting clusters have a similar structure to the clusters in 2.1.2 with an empty class, metallic and matt surface classes. Using the technique described in 2.1.3, the newly found clusters and the semi-constrained Gestalt codes, a colour axis could be found but no shape axis.
It can be concluded that more regularisation results in better separated and more meaningful full clusters and that an unconstrained Gestalt code space has no advantages but rather more disadvantages.

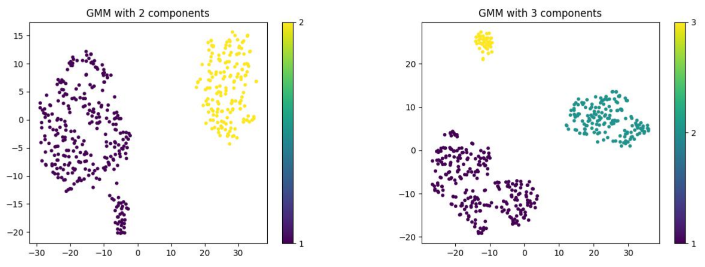
2.2.2 Categorical distribution

In addition to the two previous methods, another regularisation technique inspired by Dreamer V2 was used. There, the authors were able to improve the performance by replacing the Gaussian latents with categorical variables. They use 32 categorical variables with 32 classes each [6]. Since the resulting Gestalt code is calculated deterministically rather than stochastically, it must be transformed into a categorical distribution. Therefore, the Gestalt code is first transformed into a fixed set of categorical variables and corresponding classes. To meet the requirements of a probability distribution, the softmax function is then applied to each categorical variable independently.
Since the size of the Gestalt code affects the number of categorical variables and resulting classes for each variable, two different sizes of Gestalt code were used, a 96 dimensional Gestalt code with 8 categorical variables with

12 classes each and a 192 dimensional Gestalt code with two different settings. One has 16 categorical variables with 12 classes each, the other has 24 categorical variables with 8 classes each. In order to obtain comparable results, the number of parameters of Loci must remain constant in all experiments. By doubling the dimensionality of the Gestalt code, the parameters of the encoder increase from 835k to 1.1 million. To compensate, the heads and layers of the transition module were halved, reducing the parameters from 1.1 million to 842k. Although this results in a small change in the parameter distribution between the different modules, it is the best way to achieve results that are as comparable as possible.

In the previous experiments, the effect of the different Gestalt code sizes and regularisations was determined by the changes in the clusters and found colour and shape axes. To check how the different categorical distributions affect the results of the CATER-tracking challenge, the average top 1 accuracy of Snitch detection is also compared across the different settings. This provides a more objective way of comparing the different settings.

The top 1 accuracy for the 96 dimensional Gestalt code is 81%, for the 192 dimensional Gestalt code with 16 categorical variables it is 79% and for 24 variables it is 36%. As the last option gives much worse results, it is not analysed further. The clusters for the remaining two settings are well separated but also widely spread (Figure 11). This supports the assumption that good performance in the CATER-tracking challenge requires that the Gestalt codes are on a data manifold. It can be concluded that the



(a) 8 categorical variables with 12 classes

(b) 16 categorical variables with 12 classes

Figure 11: Gaussian Mixture Model clusters of the categorical Gestalt codes, extracted after the transition module, after dimensionality reduction with t-SNE using the CATER dataset

categorical distribution Gestalt codes, with the correct settings, also give good results and open up new possibilities for regularisation, which will be explored in the following.

2.2.3 Mutual information

Since the rapid rise of deep neural network architectures with a bottleneck layer that forces a low-dimensional representation, called a latent code, of the data to be learned, such as the Gestalt codes in the Loci architecture, the question of the importance of a disentangled latent code has arisen and has been investigated in several papers. In one paper, the authors were able to achieve good results in the creation of novel faces containing features from several known images, by emphasising a disentangled latent code via independent subspaces [3]. Therefore, what follows is an examination of the effects of regularising the categorical distribution Gestalt code to produce independent variables subspaces.

To create independent variables, the mutual information between the variables is used. In information theory, the mutual information between X and Y, $I(X;Y)$, measures the "amount of information" learned from the knowledge of the random variable Y about the other random variable X. There is an intuitive interpretation: $I(X;Y)$ is the reduction of uncertainty in X when Y is observed. If X and Y are independent, then $I(X;Y) = 0$, because knowing one variable tells us nothing about the other. By reducing the mutual information between the categorical variables in the Gestalt code, they should become independent.

To enforce this constraint on Loci, an additional regularisation loss is introduced. The *mutual information loss* is estimated by calculating the mutual information between all categorical variables of the Gestalt codes over a batch using the Nonparametric Entropy Estimation Toolbox [11].

For the experiments carried out, the 96-dimensional Gestalt code with 8 variables and 12 possible classes is used, since this setting previously gave the best results. Since the *mutual information loss* is a regularisation loss, it must be multiplied by a factor smaller than one before being added to the combination of losses given in [12], otherwise the performance will be negatively influenced. Two different factors were used during the experiments, 1×10^{-4} and 1×10^{-2}.The top 1 accuracy on the CATER-tracking challenge for both runs using the standard training procedure given in 2.2 is 80%. So this regularisation technique did not improve performance.

It can be concluded that the categorical Gestalt codes may have the potential to achieve similar results to the binary Gestalt codes, but as observed, the performance is sensitive to the number of categorical variables and classes used, while there is no performance increase by creating independent variables. As the binary Gestalt codes have already shown that they are able to achieve good results, this regularisation technique will be used for the rest of this research project.

2.3 Closed-loop

So far, Loci always needs an image as input to compute the image for the next time step. For the CATER-tracking challenge this is not a problem, since a continuous stream of images is provided and only the position of the snitch needs to be tracked over time. However, if Loci is used for the Moving MNIST or an intuitive physics dataset in a closed-loop, the images generated by the decoder module must be fed to the encoder module. This introduces an unnecessary overhead, since at time $t = n$ the transition module produces Gestalt codes for time $t = n+1$, which should be the same Gestalt codes that the encoder produces at $t = n+1$. If this constraint were satisfied, Loci could be run in a closed-loop with no overhead. However, as observed in 2.1.2, the Gestalt codes produced by the encoder and the transition are optimised for different purposes. While the Gestalt codes produced by the transition module must produce the correct image after processing by the decoder module, the Gestalt codes produced by the encoder module must contain information that the transition module needs to predict the next image. Therefore, a prediction module is needed to produce Gestalt codes that can be fed directly into the encoder. In this section, a sufficient prediction module is designed and implemented. It is then used to train and run Loci in a closed-loop.

2.3.1 Prediction module

The prediction module must produce Gestalt codes that are similar to the Gestalt codes produced by the encoder at $t = n + 1$, using the Gestalt codes from the transition module at $t = n$. Although the binary Gestalt codes gave good results, the pre-binarisation Gestalt codes are used as input to the prediction module in order to preserve more information. In addition, the hidden states from the transition module are also used as input to the prediction module to enrich the Gestalt codes with information that is not needed to generate images by the decoder module, but can contain important information such as the speed at which the objects are moving and how the objects will interact with each other. Two different architectures for the prediction module have been implemented and their effect on the top1 of the CATER tracking challenge has been tested, as this new module adds an extra layer of complexity to the Loci architecture, which should not reduce the overall performance.
The first architecture is a multilayer perceptron (MLP) with 3 linear layers ($496 \rightarrow 200, 200 \rightarrow 200, 200 \rightarrow 96$) and ReLU as a non-linear activation function. As input, the Gestalt code for each object and the hidden states of the transition module are concatenated into a 496-dimensional vector which is fed into the MLP, which has a total of 158k parameters. The second architecture has first a multi-head attention as described in [14], followed by

an MLP with three linear layers ($96 \rightarrow 200, 200 \rightarrow 200, 200 \rightarrow 96$) and ReLU as a non-linear activation function, with a total of 174k parameters. The sequence of 10 Gestalt codes, one for each object, is used as the query, while the sequence of hidden states from the transition module is used as key and value to implement an eight-headed Cross-Attention. For both architectures, the output of the module is multiplied by a learnable parameter, initially set to near zero, and added to the Gestalt code used as the input to the module. This learnable residual connection allows the model to learn over time to give more weight to the output of the model compared to the residual connection, making training much easier, especially at the beginning.
The loss function used was the mean square error (MSE) between the Gestalt codes produced by the prediction module and the encoder. This loss was then added to the combination of losses given in [12]. To make training feasible, this loss must be multiplied by a small factor. For the following experiments, the factor 1×10^{-6} was used. To reduce the impact of the prediction module on the overall training and architecture, the hidden states from the transition module were detached from the computational graph before being used as input to the prediction module. With this setup, the gradient from the training of the prediction module does not flow into the transition module and therefore is not influenced by the prediction module. In order to control the effect of the prediction module on the encoder, both architectures were trained with and without gestalt codes detached from the encoder using the same training procedure described in 2.2 using the CATER dataset. To compare the two architectures and configurations, the MSE and top 1 accuracy of the Snitch detection were used (Table 1).

	MSE	Snitch detection
MLP with detached Gestalt codes	0.2	78%
MLP without detached Gestalt codes	0.07	78%
Cross-Attention with detached Gestalt codes	0.1	77%
Cross-Attention without detached Gestalt codes	0.2	79%

Table 1: Comparison of MLP and Cross-Attention architectures for the prediction module using the CATER dataset

The MSE should be low to obtain Gestalt codes that can be handled well by the transition module. If the Snitch detection is still high, the effect of the newly added prediction module on the overall loci architecture and performance can be neglected. The trade-off between these two factors is best achieved by the MLP without detached Gestalt codes. Therefore, this architecture will be used for the further closed-loop experiments to be performed in the following.

2.3.2 Training in a closed-loop

Although the CATER dataset was well suited to measuring the impact of various architectural changes during this project, it is not the best option for evaluating how well Loci works in a closed-loop, as the objects in the CATER dataset are raised and lowered by a magic hand. Therefore, the CLEVRER dataset is used to evaluate the closed-loop behaviour of Loci. The CoLlision Events for Video REpresentation and Reasoning (CLEVRER) is a diagnostic video dataset for the systematic evaluation of computational models in a variety of reasoning tasks [3].
In order for Loci to run in a closed-loop, some changes had to be made to the input of the different modules. The background module now takes as input the image and mask generated by Loci and a zero tensor as the error mask, the prediction module takes the Gestalt codes before binarisation and the hidden states from the transition module. The transition module must now use the position and priority code generated by itself at the last time step and the Gestalt code from the prediction module. No changes were required to the input of the decoder module.
As a first step, two different training configurations were run. The first configuration used the standard training technique as described in 2.2, but with the prediction module integrated. In the second configuration, in 10% of the iterations, the Gestalt codes from the prediction module were used instead of those from the encoder module. Figure 12 shows the images generated when the differently trained models were operated in a closed-loop. To do this, the first image is fed into Loci for ten iterations, then Loci is operated normally for 30 time steps to introduce the objects and movements into the system, as the objects appear over time and only enter the image after several time steps. This is followed by the closed-loop phase as described above.

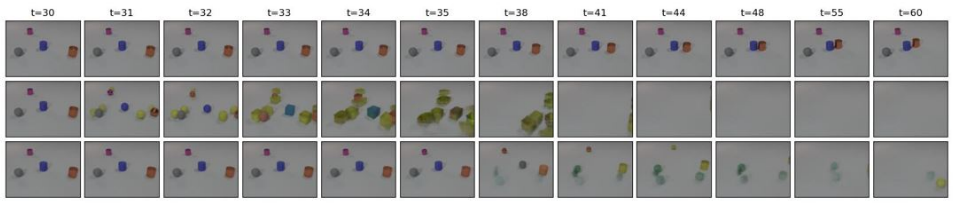


Figure 12: Closed-loop Loci, **top row** ground truth, **middle row** no closed-loop training, **bottom row** 10% training with Gestalt codes generated by the prediction module using the CLEVRER dataset

Without additional closed-loop training, new objects are created and the shape of the old objects is lost after just one time step in the closed-loop (middle row figure 12). Over time, the incorrectly created objects are pushed out of the image. With 10% closed-loop training, the shape of the

objects remains correct for 5 time steps, but the movement of the objects is not tracked correctly. Also with this setting the objects are pushed out of the image over time (bottom row figure 12). The pushing out of the objects can be seen as a shortcut learned by Loci to minimise the error. It can be concluded that the closed-loop training leads to better results, which are still not good in terms of position tracking. This could be due to the fact that there is not much movement of the objects between two consecutive frames. Until now, the parameters of Loci have been optimised after every single time step, so there has never been much dynamics introduced into the system during training. In order to improve the closed-loop performance of Loci, a new training strategy needs to be implemented that takes all these problems into account, which will be explained below.
To introduce dynamics into the system, the parameters of Loci are only updated after three time steps. Also in the first training phase, 60% of the time steps are performed in a closed loop and 70% in the second phase. The MSE loss of the prediction module is reduced by a factor of 1×10^{-4} in the second training phase compared to 1×10^{-6} in the first phase before being added to the total loss used for optimisation. To avoid making the training too complex and performing too many steps in a closed loop, the maximum number of steps performed in a closed loop during training is set to six. When Loci is not running in closed-loop, it operates in the normal way by feeding the input images into the coding module. Short-cut learning should be avoided by using the probabilistic method to determine whether the next iteration will be closed-loop, and only setting the percentage of iterations that will be closed-loop. Figure 13 shows the images generated by Loci when trained using the procedure described above.

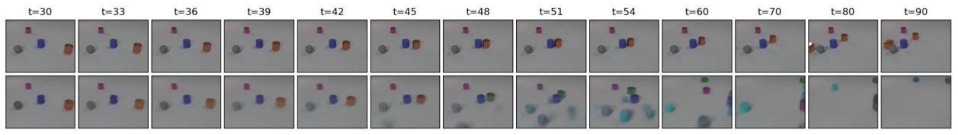


Figure 13: Loci run in a closed loop with specific closed loop training, **top row** ground truth, **bottom row** output from Loci

Loci is now able to correctly track the position of the objects while maintaining the correct shape for more than ten time steps into the future. In the following time steps, the sharp boundaries of the objects and the correct position are lost. Then, as in the previous model, new imaginary objects are created and later pushed out of the image, starting after 20 time steps. Even though this new training procedure could achieve a significant improvement, there are still some issues such as the loss of sharp boundaries and the prediction of the position. These aspects will be further investigated below.

As a first step, the current model was trained for one more closed-loop

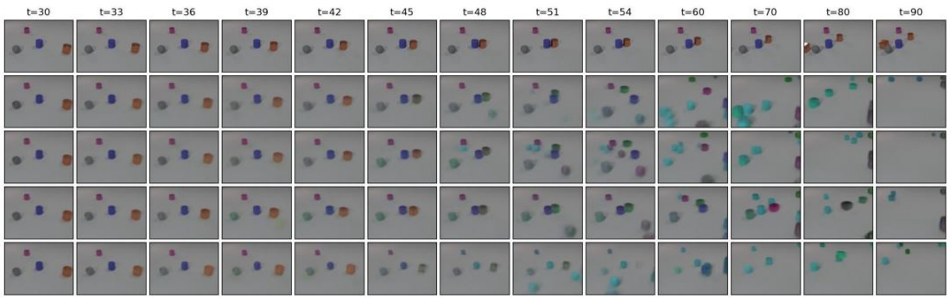


Figure 14: Loci running in closed loop with more training and using input images to generate background mask, position code and priority of objects, **first row** ground truth, **second row** output from Loci with pure closed loop, **third row** closed-loop Loci using ground truth image for background mask, **fourth row** closed-loop Loci with position codes using ground truth images, **fifth row** closed-loop Loci using ground truth image for priority

specific training phase with four iterations before optimisation, an increase in closed-loop percentage of 85%, a maximum of eight steps in closed-loop and a reduced factor of 1×10^{-3} for the loss. This longer closed-loop training improved the position tracking and it is now almost perfect up to 18 time steps into the future (second row figure 14). When the ground truth images are used to create the background mask or the priority code for the objects, there is a decrease in performance and more imaginary objects are created (third row and fifth figure 14). This could be due to the fact that most of the time the loci were not trained with the ground truth images for the background mask and priority code. When the input images are used for the position code, the positions of the objects are correctly tracked for more than 20 time steps (fourth row figure 14). This shows the importance of not only the Gestalt codes, but also the position codes. This is important if Loci is to produce good results in a closed loop. For all the different configurations, at least after 30 closed-loop steps, there is almost no similarity left between the ground truth and the output produced by Loci, and after 60 time steps most objects are pushed out of the frame.
It can be concluded that the prediction module plays a crucial role for a good closed-loop performance as well as the correct training procedure. For 20 time steps Loci can produce good results in a closed loop, but for long runs like 30 steps and more the performance drops drastically. This could also be due to the dataset, as in the CLEVRER dataset new objects are pushed in and out of the images, making long-term prediction very complex.

3 Conclusion and future work

The topics covered in this research project consisted of three parts. In the first part, a data analysis pipeline for the Gestalt codes was implemented. This pipeline was used to investigate the differences between the different versions of the Gestalt codes. It was found that the clusters of Gestalt codes should be well separated to achieve good performance in the CATER-tracking challenge. It was also possible to find disentangled shape and colour axes for the binary Gestalt codes. This analysis provided insight into why the binary Gestalt codes are superior to the continuous version. These findings, in combination with the implemented pipeline, can be used in the future to further develop Loci, while keeping an emphasis on the Gestalt codes to meet the requirements.
In the second phase of the project, new regularisation techniques were applied to the Gestalt code and the resulting Gestalt codes were analysed using the previously implemented pipeline. Transforming the Gestalt code into a categorical distribution opens up new possibilities. However, even when the mutual information between the categorical variables in a Gestalt code is explicitly reduced, the categorical Gestalt codes cannot outperform the binary Gestalt codes in the CATER-tracking challenge, which was used as a performance indicator.
In the third part of the research project, a prediction module was designed, implemented and tested to run Loci in a closed loop using only the latent Gestalt and position code, thereby reducing the overhead of generating images for a closed loop. Using a modified closed-loop training protocol, Loci was able to correctly predict almost 20 frames into the future of the CLEVRER dataset. The trained closed loop should be tested on other datasets to investigate whether the problem of creating imaginary objects and pushing out other objects is due to the dataset or to the model and training. Simply using the position codes from the transition module as input to the transition module for the next time step does not yield the ground truth position, especially over time. This highlights the importance of both the what and the where components of the binding problem. Further work should be done to improve the position tracking using the position codes and thereby improve the already good closed-loop performance. Then, as a video reasoning system, Loci will be able to model the properties of physical objects over time into the future and continue to contribute to overcoming the shortcomings in this area of research.

4 References

[1] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[2] Ira Assent. Clustering high dimensional data. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(4):340–350, 2012.

[3] Maren Awiszus, Hanno Ackermann, and Bodo Rosenhahn. Learning disentangled representations via independent subspaces. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[4] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, 29, 2016.

[5] Rohit Girdhar and Deva Ramanan. CATER: A diagnostic dataset for Compositional Actions and TEmporal Reasoning. In *ICLR*, 2020.

[6] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] Pavlin G. Poličar, Martin Stražar, and Blaž Zupan. opentsne: a modular python library for t-sne dimensionality reduction and embedding. *bioRxiv*, 2019.

[9] Douglas A Reynolds et al. Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663), 2009.

[10] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[11] Greg Ver Steeg. Non-parametric entropy estimation toolbox. `https://github.com/gregversteeg/NPEET`, 2023. Accessed on March 3, 2023.

[12] Manuel Traub, Sebastian Otte, Tobias Menge, Matthias Karlbauer, Jannik Thümmel, and Martin V Butz. Learning what and where–unsupervised disentangling location and identity tracking. *arXiv preprint arXiv:2205.13349*, 2022.

[13] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.