Complex-Valued Autoencoders for Object Discovery

Sindy Löwe
UvA-Bosch Delta Lab
University of Amsterdam
loewe.sindy@gmail.com

Phillip Lippe
QUVA Lab
University of Amsterdam

Maja Rudolph
Bosch Center for AI

Max Welling
UvA-Bosch Delta Lab
University of Amsterdam

Abstract

Object-centric representations form the basis of human perception and enable us to reason about the world and to systematically generalize to new settings. Currently, most machine learning work on unsupervised object discovery focuses on slot-based approaches, which explicitly separate the latent representations of individual objects. While the result is easily interpretable, it usually requires the design of involved architectures. In contrast to this, we propose a distributed approach to object-centric representations: the Complex AutoEncoder. Following a coding scheme theorized to underlie object representations in biological neurons, its complex-valued activations represent two messages: their magnitudes express the presence of a feature, while the relative phase differences between neurons express which features should be bound together to create joint object representations. We show that this simple and efficient approach achieves better reconstruction performance than an equivalent real-valued autoencoder on simple multi-object datasets. Additionally, we show that it achieves competitive unsupervised object discovery performance to a SlotAttention model on two datasets, and manages to disentangle objects in a third dataset where SlotAttention fails – all while being 7-70 times faster to train.

1 Introduction

Object discovery plays a crucial role in human perception. It allows us to interact seamlessly with our environment, reason about it, and to generalize systematically to new settings. To achieve this, our brains have overcome the binding problem (Greff et al., 2020): even though biological neural networks exhibit (relatively) fixed connections, they can flexibly and dynamically bind information belonging to separate entities.

Currently, most work dedicated to solving the binding problem in machine learning focuses on slot-based approaches (Hinton et al., 2018; Burgess et al., 2019; Greff et al., 2019; Lin et al., 2020; Locatello et al., 2020; Kipf et al., 2021). Here, the latent representations are explicitly separated into "slots" which learn to represent different objects. These slots are highly interpretable; however, the introduction of a separate object-centric representation module in a model that otherwise does not exhibit object-centric features causes a number of problems. For one, it usually requires the design of involved architectures with iterative procedures, elaborate structural biases, and intricate training schemes to achieve a good separation of object features into slots. Moreover, this separation is often achieved by limiting the information flow and expressiveness of the model, leading to failure cases for complex objects, e.g. with textured surfaces (Karazija et al., 2021). Finally, since all slots are created at the same level of representation, this approach cannot inherently represent part-whole hierarchies.

Preprint.

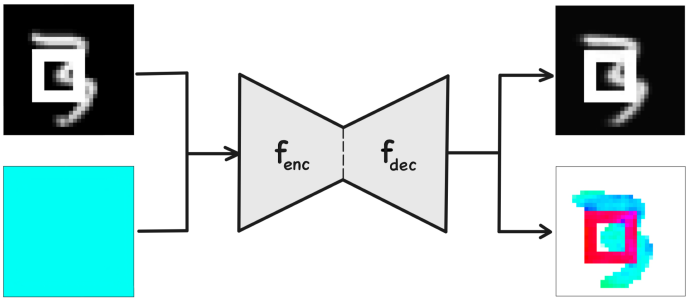

Figure 1: We propose the Complex AutoEncoder (CAE) – a simple and efficient object discovery approach leveraging complex-valued activations in an autoencoding architecture. Given a complex-valued input whose magnitude represents the input image and whose phase is set to a fixed value, the model learns to reconstruct the input image, and to represent the disentangled object identities in its phase values in an unsupervised way.

To overcome these issues of slot-based approaches, we take inspiration from the temporal correlation hypothesis from neuroscience (Singer, 2009) and design a model that learns representations of objects that are distributed across and embedded in the entire architecture. The temporal correlation hypothesis describes a coding scheme used by biological neurons to overcome the binding problem. Essentially, it posits that each neuron sends two types of messages: (1) whether a certain feature is present or not, encoded by the discharge frequency or rate code, and (2) which other neurons to bind information to, encoded by the synchronicity of firing patterns.

Following Reichert & Serre (2014), we abstract away these two messages requiring binary spikes and temporal dynamics by making use of complex-valued activations in artificial neural networks. This allows each neuron to represent the presence of a feature through the complex number's magnitude, and to bind this feature to other neurons' features by aligning its phase value with theirs. Reichert & Serre (2014) applied this coding scheme to a deep Boltzmann machine trained with real-valued activations to create object-centric representations at test-time, which required 10s-100s of iterations to settle to an output configuration. In contrast to this, we apply this coding scheme to a modern convolutional autoencoder, and train this model end-to-end with complex-valued activations. Through a careful setup of each layer, this allows us to create phases representative of object identity within a single forward-pass through the model – significantly increasing the efficiency and practicality of our approach.

Inspired by the temporal correlation hypothesis, we propose the Complex AutoEncoder (CAE, Figure 1), an object discovery model that leverages complex-valued activations in a standard autoencoder. In the CAE, we use the magnitude of the complex-valued reconstructions to train the model using a standard mean squared error loss, and the phase values for the unsupervised creation of pixel-accurate segmentation masks. Overall, this leads to an approach in which the object-centric representations are embedded in and distributed across the entire architecture. Our contributions are as follows:

- We show how the coding scheme described by the neuroscientific temporal correlation hypothesis can be applied to modern deep artificial neural networks in an efficient and effective way by introducing the Complex AutoEncoder (CAE).
- We show that the CAE creates more accurate reconstructions than its real-valued counterpart on simple multi-object datasets, indicating that it makes effective use of its complex-valued activations to represent different objects.
- We show that the CAE achieves competitive or substantially better object discovery performance on simple multi-object datasets compared to SlotAttention (Locatello et al., 2020), a state-of-the-art slot-based approach, while being 7-70 times faster to train.

2 The Temporal Correlation Hypothesis

The Complex AutoEncoder takes inspiration from neuroscience, where the temporal correlation hypothesis describes a possible mechanism underlying object-centric representations in the brain. In

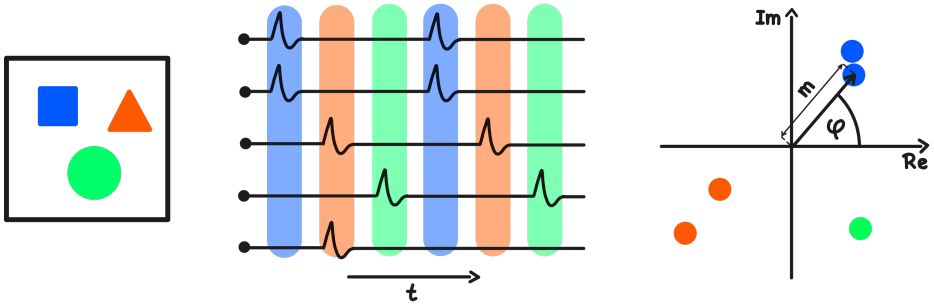

Figure 2: The temporal correlation hypothesis. **Left:** Input image with different objects. **Middle:** Implementation of the temporal correlation hypothesis with spiking neurons. Their spiking rate represents the presence of a feature, while their synchronization represents which features should be bound together to jointly represent an object. **Right:** Implementation of the temporal correlation hypothesis with complex-valued activations. Each complex number $z = m \cdot e^{i\varphi} \in \mathbb{C}$ is defined by its magnitude m and phase φ. This allows for an equivalent representation of feature presence and synchronization through the magnitude and phase values, respectively.

this section, we will outline this hypothesis, and draw a connection to the complex-valued activations implemented in our proposed model.

In neuroscience, the binding problem describes the open question of how the brain binds information flexibly and dynamically within a network of fixed connectivity and segregated processing areas to give rise to coherent percepts, e.g. for different objects. Only by overcoming the binding problem, the brain is capable to represent all manner of objects, to attain a compositional understanding of the world, and to generalize robustly to new environments. While there is an ongoing debate as to their functional importance (Shadlen & Movshon, 1999; Ray & Maunsell, 2010), various works posit that the brain uses temporal dynamics to overcome the binding problem (Milner, 1974; Von Der Malsburg & Schneider, 1986; Singer & Gray, 1995; Engel et al., 1997; Singer, 1999; Fries, 2005; Singer, 2009; Palmigiano et al., 2017). Essentially, these theories postulate that the brain binds information from different neurons by synchronizing their firing patterns, while desynchronized firing represents information that ought to be processed separately. There are various manifestations of this theory; in this work, we will focus on the temporal correlation hypothesis as developed by Singer & Gray (1995); Singer (2009).

The temporal correlation hypothesis describes how the oscillating behavior of biological neurons could be used by the brain to overcome the binding problem. It posits that each neuron sends two messages through its spiking pattern (Figure 2 - Middle): (1) The discharge frequency or rate code of a neuron encodes whether the feature that it is tuned to is present or not. The real-valued activation of neurons in artificial neural networks can be interpreted to be the technical implementation of this message. (2) The relative timing between two neurons' spikes encodes whether the represented features of these neurons should be bound together or not. When firing in synchrony, the features they represent will be evaluated jointly by the target neuron and are thus bound together in a flexible and dynamic way. Currently, very few works explore the use of this second message type in artificial neural networks.

There is a broad range of neuroscientific evidence in support of the temporal correlation hypothesis. For one, oscillating neural activity (i.e. brain waves) is observed across the brain. Gamma waves in the frequency range between 30 and 60Hz, in particular, occur when paying focussed attention (Bear et al., 2020). Additionally, their synchronization probabilities correlate well with Gestalt Principles (Engel et al., 1991; Castelo-Branco et al., 2000), which describe the principles that humans employ to group perceptual patterns into jointly perceived objects. For example, regular gratings invoke a maximal synchronization probability, while random dot patterns reduce this probability to its minimum (Engel et al., 2001). Additionally, circumstantial evidence points toward the coding scheme described by the temporal correlation hypothesis as a valid mechanism: both this coding scheme and established learning mechanisms such as Hebbian Learning (Hebb, 1949) and spike-time dependent plasticity (STDP) (Caporale & Dan, 2008) require neurons to be very sensitive to the precise timing of spikes. Plus, the temporal correlation hypothesis induces a natural upper limit on the number of

objects that can be represented within a single oscillatory cycle, which goes in line with the capacity restrictions observed in human short-term memory (Cowan, 2001).

In this paper, we take inspiration from the temporal correlation hypothesis to develop a machine learning approach capable of overcoming the binding problem. Inspired by previous work (Reichert & Serre, 2014), we abstract away from the spiking nature of biological neural networks and instead represent the two message types described above with the help of complex numbers (Figure 2 - Right). As a result, we create an artificial neural network with complex-valued activations $z = m \cdot e^{i\varphi} \in \mathbb{C}$ in which the magnitude m can be interpreted as the rate code emitted by a spiking neuron (message (1) above) and the phase φ can be used as the mathematical mechanism to capture the temporal alignment of the firing patterns (message (2) above). In the next section, we will describe the setup which achieves this.

3 Complex AutoEncoder

We propose the Complex AutoEncoder (CAE) – a model for object discovery that leverages mechanisms inspired by the temporal correlation hypothesis to create distributed object-centric representations. We start by injecting complex-valued activations into a standard autoencoding architecture (Section 3.1). Ultimately, we want these complex-valued activations to convey two messages: their magnitudes should represent whether a feature is present, and their phase values should represent which features ought to be bound together. In Sections 3.2 and 3.3, we describe the setup of the model that gives rise to this coding scheme. Using the described mechanisms, after unsupervised training on a multi-object dataset, CAE's phase values represent different objects in a scene. In Section 3.4, we describe how we evaluate these phase values to produce object-wise representations, as well as pixel-accurate segmentation masks for object discovery.

3.1 Complex-Valued Activations in Autoencoders

To enable an autoencoder to develop object-centric representations, we inject it with complex-valued activations. In this section, we will describe how we translate between the real-valued inputs and outputs used for training the model and the complex-valued activations used for representing object-centric features.

The Complex AutoEncoder (Figure 1) takes a positive, real-valued input image $\mathbf{x} \in \mathbb{R}^+$ and associates each pixel with an initial phase $\boldsymbol{\varphi} = 0 \in \mathbb{R}$ to create the complex-valued input $\mathbf{x}'$ to the model:

$$\mathbf{x}' = \mathbf{x} \cdot e^{i\boldsymbol{\varphi}} \quad \in \mathbb{C} \tag{1}$$

CAE applies a convolutional encoder f_{enc} and decoder f_{dec} with real-valued parameters $\boldsymbol{\theta} \in \mathbb{R}$ to this complex-valued input and creates a complex-valued reconstruction $\hat{\mathbf{z}}$:

$$\hat{\mathbf{z}} = f_{\text{dec}}(f_{\text{enc}}(\mathbf{x}')) \quad \in \mathbb{C} \tag{2}$$

To make use of existing deep learning frameworks, we do not apply layers to their complex-valued inputs directly. Instead, each layer extracts real-valued components (the real and imaginary part, or the magnitude and phase) from its input and processes them separately, before combining the results into a complex-valued output. We will describe this process in more detail in the following section.

We create the real-valued reconstruction $\hat{\mathbf{x}}$ by applying f_{out}, a 1×1 convolutional layer with a sigmoid activation function, on the magnitudes of the complex-valued output $\hat{\mathbf{z}}$ of the decoder: $\hat{\mathbf{x}} = f_{\text{out}}(|\hat{\mathbf{z}}|) \in \mathbb{R}^+$. This allows the model to learn an appropriate scaling and shift of the magnitudes to better match the input values. The model is trained by comparing this reconstruction to the original input using a mean squared error loss function $\mathcal{L} = \text{MSE}(\mathbf{x}, \hat{\mathbf{x}}) \in \mathbb{R}^+$ and by using the resulting gradients to update the model parameters.

Finally, we interpret the phase values $\boldsymbol{\varphi} = \arg(\mathbf{z}) \in [0, 2\pi)$ of the complex-valued activations $\mathbf{z} \in \mathbb{C}$ as object assignments – either to extract object-wise representations from the latent space or to obtain a pixel-accurate segmentation mask in the output space. Here, $\arg(\mathbf{z})$ describes the angles between the positive real axis and the lines joining the origin and each element in $\mathbf{z}$. In the next section, we will describe the mechanisms that encourage the model to learn phase values that are representative of object identity.

3.2 Phase Alignment of Complex Numbers

For the CAE to accomplish good object discovery performance, the phases of activations representing the same object should be synchronized, while activations induced by different objects should be desynchronized. To achieve this, we need to enable the network to assign the same phases to some activations and different phases to others, and to precisely control phase shifts throughout the network. We achieve this by following three steps for each network layer $f_{\boldsymbol{\theta}} \in \{f_{\text{enc}}, f_{\text{dec}}\}$ parameterized by $\boldsymbol{\theta} \in \mathbb{R}$ and applied to the input to that layer $\mathbf{z} \in \mathbb{C}$:

Synchronization First, we need to encourage the network to synchronize the phase values of features that should be bound together. This property is achieved naturally through the use of complex numbers: when adding two complex numbers of opposing phases, they suppress one another or even cancel one another out (a.k.a. destructive interference). Thus, to preserve features, the network needs to align their phase values (a.k.a. constructive interference).

Desynchronization Next, we need a mechanism that can desynchronize the phase values. Again, this is achieved naturally through the use of complex numbers: when adding two complex numbers with a phase difference of $90°$, for example, the result will lie in between these two numbers and thus be shifted, i.e. desynchronized by $45°$. On top of this inherent mechanism, we add a second mechanism that lends the network more control over the precise phase shifts. Specifically, we apply the weights $\mathbf{w} \in \boldsymbol{\theta}$ of each layer separately to the real and imaginary components of its input:

$$\boldsymbol{\psi} = f_{\mathbf{w}}(\mathbf{z}) = f_{\mathbf{w}}(\text{Re}(\mathbf{z}) + \text{Im}(\mathbf{z}) \cdot i) = f_{\mathbf{w}}(\text{Re}(\mathbf{z})) + f_{\mathbf{w}}(\text{Im}(\mathbf{z})) \cdot i \quad \in \mathbb{C} \tag{3}$$

Next, we add separate biases $\boldsymbol{b_m}, \boldsymbol{b_\varphi} \in \boldsymbol{\theta}$ to the magnitudes and phases of the resulting complex-valued representations $\boldsymbol{\psi}$ to create the intermediate magnitude $\boldsymbol{m_\psi}$ and phase $\boldsymbol{\varphi_\psi}$:

$$\begin{aligned} \boldsymbol{m_\psi} &= |\boldsymbol{\psi}| + \boldsymbol{b_m} && \in \mathbb{R} \\ \boldsymbol{\varphi_\psi} &= \arg(\boldsymbol{\psi}) + \boldsymbol{b_\varphi} && \in \mathbb{R} \end{aligned} \tag{4}$$

This formulation allows the bias $\boldsymbol{b_\varphi}$ to influence the phase value of each activation directly. Inherently, this enables the model to learn explicit phase shifts throughout the network and to break the symmetry created by the equal phase initialization (Equation (1)).

Desynchronization vs. Inhibition Finally, we add a mechanism that enables the model to distinguish inhibitory inputs with aligned phases from excitatory inputs with opposing phases. To illustrate: given weights $\mathbf{w}$ and activations $\mathbf{a}$, in the network formulation above it holds that $(-\mathbf{w}) \cdot \mathbf{a} = \mathbf{w} \cdot (-\mathbf{a})$. However, in our case, this is not desirable. To enable the model to learn meaningful phase shifts, it needs to be able to distinguish a negative weight from a negative activation, because one has an aligned phase and the other does not. Taking inspiration from Reichert & Serre (2014), we achieve this by taking the absolute value of the activations $(-\mathbf{w}) \cdot |\mathbf{a}| \neq \mathbf{w} \cdot |(-\mathbf{a})|$. Thus, we additionally apply each layer to the magnitude of its input and combine the result with $\boldsymbol{m_\psi}$ to create the intermediate values $\boldsymbol{m_z}$:

$$\begin{aligned} \boldsymbol{\chi} &= f_{\mathbf{w}}(|\mathbf{z}|) + \boldsymbol{b_m} && \in \mathbb{R} \\ \boldsymbol{m_z} &= 0.5 \cdot \boldsymbol{m_\psi} + 0.5 \cdot \boldsymbol{\chi} && \in \mathbb{R} \end{aligned} \tag{5}$$

3.3 Complex-Valued Activation Function

We propose a new activation function for complex-valued activations to further ensure maximal control of the network over all phase shifts. To create a layer's final output $\mathbf{z}'$, we apply a non-linearity on the magnitudes $\boldsymbol{m_z}$, but keep the phases $\boldsymbol{\varphi_\psi}$ unchanged:

$$\mathbf{z}' = \text{ReLU}(\text{BatchNorm}(\boldsymbol{m_z})) \cdot e^{i\boldsymbol{\varphi_\psi}} \quad \in \mathbb{C} \tag{6}$$

There are several things to note in this setup. First, $\boldsymbol{m_z}$ might contain negative values as a result of the summation with potentially negative values in the magnitude bias $\boldsymbol{b_m}$ (Equation (4)), as well as potentially negative values in $\boldsymbol{\chi}$ (Equation (5)). Nonetheless, it is biased to be positive due to the usage of absolute values throughout each layer. Second, by applying BatchNormalization (Ioffe & Szegedy, 2015), we ensure that – at least initially – $\boldsymbol{m_z}$ becomes zero-centered and therefore makes use of the non-linear part of the ReLU activation function. At the same time, BatchNormalization provides the flexibility to learn to shift and scale these values if appropriate. Finally, the ReLU non-linearity ensures that the magnitude of $\mathbf{z}'$ is positive and thus prevents any phase flips.

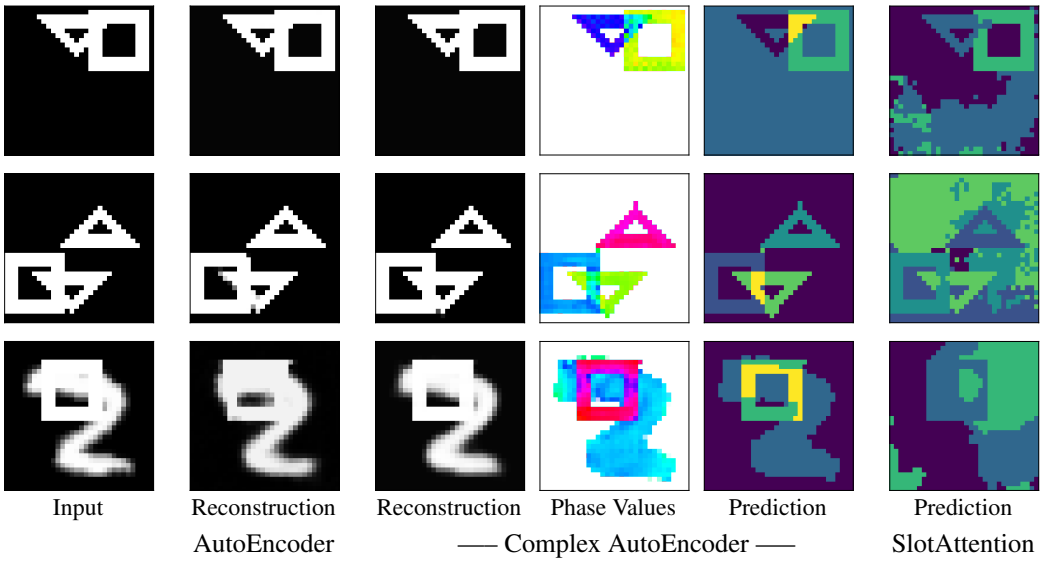

Figure 3: Visual comparison of the performance of the Complex AutoEncoder, a corresponding real-valued autoencoder and a SlotAttention model on random samples from the test sets of the 2Shapes (**Top**), 3Shapes (**Middle**) and MNIST&Shape (**Bottom**) datasets. Areas in which objects overlap are removed before applying k-means on the phase values of the Complex AutoEncoder, resulting in yellow areas in the predictions which are not evaluated. The Complex AutoEncoder produces accurate reconstructions and object separations. Note, that the model assigns small magnitudes to the background, leading to white areas in the phase images and a clear foreground/background separation in the predictions.

3.4 Creating Discrete Object Assignments from Continuous Phase Values

To evaluate the object-centricity of the complex-valued activations of the CAE, we create discrete object assignments for each feature by applying a clustering procedure to the phase values. This allows us to extract object-wise representations from the latent space, as well as pixel-accurate segmentation masks from the complex-valued reconstructions. We group the phase values into k clusters, where k corresponds to the number of objects in the input plus one for the background. Note that this is only required for the evaluation of the CAE. During training, CAE learns continuous object assignments through its phase values and therefore does not require k to be set in advance.

Before applying k-means to cluster the phase values, we apply two pre-processing steps. First, we account for the circular nature of the phase values by mapping them onto a unit circle. This prevents values close to 0 and 2π from being assigned to different clusters despite representing similar angles. Then, we scale features by a factor of $10 \cdot m$ if their corresponding magnitude $m < 0.1$, to account for the fact that the phase values of complex numbers with small magnitudes become increasingly random. As a result of this scaling, these features will fall within the unit circle, close to the origin. In our experiments, we find that they tend to be assigned their own cluster and usually represent the background. Finally, we apply k-means and interpret the resulting cluster assignment for each phase value as the predicted object assignment.

4 Results

In this section, we evaluate whether the Complex AutoEncoder can learn to create meaningful phase separations representing different objects in an unsupervised way. We will first describe the general setup of our experiments, before investigating the results across three simple multi-object datasets.

Table 1: MSE and ARI scores (mean ± standard error across 8 seeds) for the Complex AutoEncoder, its real-valued counterpart (AutoEncoder) and a SlotAttention model in three simple multi-object datasets. The proposed Complex AutoEncoder achieves better reconstruction performance than its real-valued counterpart on all three datasets. Additionally, its object discovery performance is competitive to SlotAttention on the 2Shapes and 3Shapes datasets while requiring 5-50 times fewer training steps. Finally, the CAE manages to disentangle the objects in the MNIST&Shape dataset, where SlotAttention fails.

Dataset	Model	Steps	MSE ↓	ARI+BG ↑	ARI-BG ↑
2Shapes	Complex AutoEncoder	10000	**3.322e-04** ± 1.583e-06	**0.999** ± 0.000	**1.000** ± 0.000
	AutoEncoder	10000	5.565e-04 ± 2.900e-04	–	–
	SlotAttention	500000	1.419e-04 ± 1.410e-04*	0.812 ± 0.081	**1.000** ± 0.000
3Shapes	Complex AutoEncoder	100000	**1.313e-04** ± 2.020e-05	**0.976** ± 0.002	**1.000** ± 0.000
	AutoEncoder	100000	8.568e-04 ± 9.878e-05	–	–
	SlotAttention	500000	1.827e-04 ± 3.125e-05*	0.107 ± 0.008	0.997 ± 0.001
MNIST&Shape	Complex AutoEncoder	10000	**3.185e-03** ± 1.514e-04	**0.783** ± 0.004	**0.971** ± 0.011
	AutoEncoder	10000	5.792e-03 ± 5.553e-04	–	–
	SlotAttention	500000	5.438e-03 ± 1.607e-04*	0.047 ± 0.013	0.089 ± 0.028

*The reconstruction performance of SlotAttention is not comparable due to the use of a different autoencoding setup.

4.1 Setup

Datasets We evaluate the Complex AutoEncoder on three grayscale datasets: 2Shapes, 3Shapes, and MNIST&Shape. For each of these datasets, we generate 50,000 training images and 10,000 images for validation and testing, respectively. All images contain 32×32 pixels. The 2Shapes dataset represents the easiest setting, with two randomly placed objects ($\square$, $\triangle$) in each image. The 3Shapes dataset contains a third randomly placed object ($\bigtriangledown$) per image. This creates a slightly more complex setting due to the higher object count, the two similar shapes ($\triangle$, $\bigtriangledown$), and stronger overlap between objects. Finally, the MNIST&Shape dataset combines an MNIST digit (LeCun et al., 2010) and a randomly placed shape ($\square$ or $\bigtriangledown$) in each image. This creates a challenging setting with more diverse objects. For evaluating the object discovery performance, we generate pixel-accurate segmentation masks for all images. More details in Appendix A.1.

Model & Training We make use of a fairly standard convolutional autoencoder architecture, as presented in Lippe (2021) (details in Appendix A.1). We train the model using Adam (Kingma & Ba, 2015) and a batch-size of 64 for 10,000 – 100,000 steps depending on the dataset. Within the first 500 steps of training, we linearly warm up the learning rate (Goyal et al., 2017) to its final value of $1e{-}3$. All experiments are implemented in PyTorch (Paszke et al., 2019) and were run on a single Nvidia GTX 1080Ti. To ensure the comparability of run-times between models, all experiments were run on the same machine and with the same underlying training and data-loading scripts.

Baselines We compare the CAE to two baseline models. First, we compare it against its real-valued counterpart. This autoencoder uses the same general architecture and training procedure, but does not employ complex-valued activations or any of the mechanisms described in Section 3.2. It does, however, apply BatchNormalization before each ReLU as we have found that this improves performance in the real domain as well.

The second model we compare against is an autoencoding architecture with SlotAttention (Locatello et al., 2020). SlotAttention is an iterative attention mechanism that produces k slots which learn to represent individual objects in the input. This model has achieved impressive unsupervised object discovery results on simple 2D and 3D datasets, and was recently shown to also perform well on more realistic textured video datasets in a weakly-supervised setting (Kipf et al., 2021). The details for the implementation of this model are in Appendix A.1.

Metrics We use three metrics to evaluate and compare the performance of the CAE against the baselines. We measure the reconstruction performance in terms of mean squared error (MSE). To assess the object discovery performance, we compute Adjusted Rand Index (ARI) scores (Rand, 1971; Hubert & Arabie, 1985). ARI measures clustering similarity, where a score of 0 indicates chance level and a score of 1 indicates a perfect match. We utilize ARI in two ways. First, following

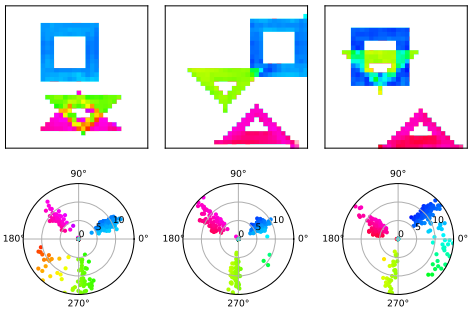

Figure 4: Phase separation in the CAE. **Top**: Output phase images. **Bottom**: Plotting every output value in the complex plane and applying the same color coding as above. The phases of the three objects are almost maximally misaligned. Interestingly, areas in which the objects overlap get assigned intermediate phase values.

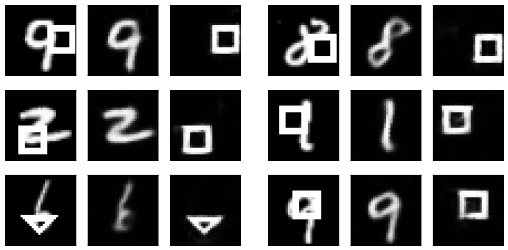

Figure 5: Investigating object-centricity of the latent features in the CAE. Columns 1 & 4: input images. Columns 2-3 & 5-6: object-wise reconstructions. By clustering features created by the encoder according to their phase values, we can extract representations of the individual objects and reconstruct them separately.

previous work (Greff et al., 2019; Locatello et al., 2020), we evaluate "ARI-BG" where we exclude the background labels from the evaluation. Additionally, we assess "ARI+BG" which evaluates the performance on all pixels. One thing to note here is that a model can achieve relatively high ARI+BG scores by assigning all pixels to the same label – thus, this score is only meaningful in conjunction with a high ARI-BG score to ensure good object separation. For both ARI scores, we remove areas in which objects overlap from the evaluation, as they are ambiguous in the grayscale setting.

4.2 Evaluation

First, we compare the quantitative performance of CAE against the two baselines in Table 1.

Reconstruction Performance The *Complex AutoEncoder* achieves a better reconstruction than its real-valued counterpart (*AutoEncoder*) on all three multi-object datasets. This illustrates that the CAE makes effective use of its complex-valued activations to create a better disentanglement of object features.

Object Discovery Performance When comparing the object discovery performance of the *Complex AutoEncoder* against *SlotAttention*, we make three observations: (1) Both models achieve (near) perfect ARI-BG scores on the 2Shapes and 3Shapes datasets. (2) On all datasets, CAE achieves considerably better ARI+BG scores indicating a more accurate separation of foreground and background. (3) On the MNIST&Shape dataset, the Complex AutoEncoder achieves an almost perfect ARI-BG score, while SlotAttention's performance is close to chance level. Overall, this shows that the Complex AutoEncoder achieves strong object discovery performance. On top of this, despite its simple and efficient design, it can overcome the challenges set by the MNIST&Shape dataset (high diversity in object shapes and relatively large object sizes), while SlotAttention cannot.

Object-Centric Representations To evaluate whether the CAE creates object-centric representations throughout the model, we apply the clustering procedure as described in Section 3.4 on the latent features created by the encoder. Then, we input the individual clusters into the decoder and fine-tune it to reconstruct separate objects. As shown in Figure 5, this procedure allows us to create accurate reconstructions of the individual objects. This indicates that the phase values are representative of object identity throughout the model, and that they allow us to extract object-wise representations in an unsupervised way.

Training Time As can be seen from Table 1, CAE requires between 10,000 and 100,000 training steps to achieve these results. SlotAttention, on the other hand, requires 500,000 training steps (see Figure 6 in the Appendix for training curves). Additionally, each training step of SlotAttention takes on average 1.4 times longer compared to a training step of the Complex AutoEncoder, due to its

iterative attention procedure as well as additional architectural details required to make the model work. Overall, this results in a training time of just under 8 minutes for the Complex AutoEncoder when trained for 10,000 steps and over 9 hours for SlotAttention. Thus, depending on the dataset, CAE can be trained between 7-70 times faster than SlotAttention.

Qualitative Evaluation In Figure 3, we show exemplary outputs of the three compared models for each dataset (for more results, see Appendix A.2). These outputs highlight the accurate reconstructions and object separations that the Complex AutoEncoder produces. When looking more closely at the phase separation created by the Complex AutoEncoder as shown in Figure 4, we find that it assigns almost maximally distant phase values to the different objects. Interestingly, the phase values of overlapping areas tend to fall in between the phase values of the individual objects. Since it is ambiguous in the 3Shapes dataset, which object is in the foreground and which one is in the background, this shows that the model accurately expresses the uncertainty that it encounters in these overlapping areas.

Sensitivity Analysis We evaluate the influence of certain hyperparameters on the performance of the Complex AutoEncoder in Table 2. We find that there are several crucial components to the CAE without which it cannot achieve a meaningful phase separation: the χ term (Equation (5)) that ensures that desynchronized inputs influence a layer's output differently from inhibitive ones; applying BatchNormalization within the activation function (Equation (6)); and the use of real-valued weights instead of complex-valued ones. Additionally, we find that the final 1×1 convolutional layer f_{out} that creates the real-valued reconstruction from the magnitudes of the complex-valued outputs improves the model's performance. Finally, we observe that the bottleneck size has relatively little influence on the performance of the Complex AutoEncoder (Figure 7 in the Appendix). This indicates that the CAE does need to restrict the expressivity of the model to create disentangled object representations.

Table 2: Sensitivity Analysis on the 2Shapes dataset (mean $\pm$ standard error across 8 seeds). We find that there are several crucial components that are required to enable the Complex AutoEncoder to separate objects in its output phase values.

Name	MSE	ARI-BG
Complex AutoEncoder	3.322e-04 $\pm$ 1.583e-06	1.000 $\pm$ 0.000
$-\ f_{\text{out}}$	2.462e-03 $\pm$ 1.458e-03	0.939 $\pm$ 0.039
$-\ \chi$	3.227e-03 $\pm$ 5.013e-04	0.074 $\pm$ 0.068
– BatchNorm	6.165e-02 $\pm$ 2.228e-02	0.373 $\pm$ 0.137
Complex Weights	5.465e-04 $\pm$ 8.980e-05	0.122 $\pm$ 0.101

5 Related Work

Object Discovery There is a broad range of approaches attempting to solve the binding problem in artificial neural networks (see Greff et al. (2020) for a great overview). However, most works focus on one particular representational format for objects: slots. These slots create an explicit separation of the latent representations for different objects. Additionally, they create an explicit separation between a specialized object-centric representation part of the model and non-object-centric parts that merely support the former, for example, through encoding and decoding functionality. In contrast to this, the Complex AutoEncoder embeds object-centricity into the entire architecture and creates distributed object-centric representations.

To create slot-based object-centric representations, different mechanisms have been proposed to overcome the binding problem, i.e. to break the symmetry between the representations of different objects in a network of fixed weights. One way to break this symmetry is by enforcing an order along a certain axis. This can be achieved by imposing an order on the slots (Eslami et al., 2016; Burgess et al., 2019; Engelcke et al., 2020), by assigning slots to particular spatial coordinates (Santoro et al., 2017; Lin et al., 2020), or by learning specialized slots for different object types (Hinton et al., 2011, 2018). Approaches that create the most general slot representations do not enforce any such order, but require iterative procedures to break the symmetries instead (Greff et al., 2019; Goyal et al., 2021). SlotAttention (Locatello et al., 2020; Kipf et al., 2021), which falls into this final category, breaks the symmetry between slots through an iterative attention mechanism. It embeds this object-centric representation mechanism into an autoencoding architecture, requiring special attention on the design of the decoder (Singh et al., 2021). Alternative unsupervised training methods for SlotAttention with e.g. contrastive learning remain restricted to simplistic environments (Löwe et al., 2020).

Object Discovery with Complex-Valued Networks A variety of research has explored different activation functions, training regimes, and applications for complex-valued neural networks (see Bassey et al. (2021) for a review). Despite this, there has been little research on the use of complex-valued networks for object discovery. The earliest works in this direction are by Mozer et al. (1992); Zemel et al. (1995). Their architectures learn to assign different phase values to different objects through a supervised training procedure. The biological plausibility of this approach is reduced by the usage of complex-valued weights. Rao et al. (2008); Rao & Cecchi (2010, 2011) propose a complex-valued neural network with real-valued weights. By training their network on images of individual objects, they enable it to separate overlapping objects of the same type on the test images. This method relies on Hebbian Learning and sparse coding, which leads to a winner-takes-all dynamic: a single neuron in the highest layer gets activated for each specific object type. Finally, Reichert & Serre (2014) train a deep Boltzmann machine on datasets similar to the ones presented here. In contrast to our method, they train the network with real-valued activations and weights and inject complex-valued activations only at test time.

All these methods initialize the complex-valued inputs to their networks with the magnitude of the input images, but with random phase values. As a result, they require iterative procedures with 10s-100s of iterations to settle to an output configuration. In contrast to that, the proposed Complex AutoEncoder initializes all phase values with a fixed value and – through a careful design of its layer-wise operations – only requires a single forward-pass through the model. This greatly improves the efficiency and practicality of complex-valued neural networks for object discovery.

6 Conclusion

Summary We present the Complex AutoEncoder – an object discovery approach implementing distributed object-centric representations. Taking inspiration from the temporal correlation hypothesis from neuroscience, we develop a neural network that leverages complex-valued activations. As a result, the activation's magnitudes encode feature information and their phases encode object affiliation. We show that this approach achieves highly competitive object discovery results on simple multi-object datasets while being substantially faster to train.

Limitations and Future Work The proposed Complex AutoEncoder constitutes a first step towards efficient distributed object-centric representation learning, but some limitations remain. Most importantly, due to the limited range of the phase values, object discovery approaches using complex-valued activations can only represent a small number of objects at a time. Additionally, we find that the phase values of different objects are interdependent and thus require the same set of objects or object-types to be present in each image to create reliable phase separations. Finally, it remains unclear how complex-valued networks for object discovery could be applied to RGB images. These restrictions currently prevent these approaches from being applied to more complex multi-object datasets. Nonetheless, the Complex AutoEncoder provides an important step forward by proposing a simple and efficient non-iterative design, that, for the first time, was shown to achieve competitive results to a slot-based approach in simple multi-object datasets. It remains an intriguing direction for future research to overcome the limitations described above and to uncover the full potential of distributed object-centric representation learning approaches.

Acknowledgements

We thank Emiel Hoogeboom, T. Anderson Keller, and Joop Pascha for their valuable feedback on the manuscript.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Joshua Bassey, Lijun Qian, and Xianfang Li. A survey of complex-valued neural networks. *arXiv preprint arXiv:2101.12249*, 2021.

Mark Bear, Barry Connors, and Michael A Paradiso. *Neuroscience: Exploring the Brain, Enhanced Edition: Exploring the Brain*. Jones & Bartlett Learning, 2020.

Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. MONet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

Natalia Caporale and Yang Dan. Spike timing–dependent plasticity: a hebbian learning rule. *Annual Review of Neuroscience*, 31:25–46, 2008.

Miguel Castelo-Branco, Rainer Goebel, Sergio Neuenschwander, and Wolf Singer. Neural synchrony correlates with surface segregation rules. *Nature*, 405(6787):685–689, 2000.

Nelson Cowan. The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and brain sciences*, 24(1):87–114, 2001.

Andreas K Engel, Peter König, and Wolf Singer. Direct physiological evidence for scene segmentation by temporal coding. *Proceedings of the National Academy of Sciences*, 88(20):9136–9140, 1991.

Andreas K Engel, Pieter R Roelfsema, Pascal Fries, Michael Brecht, and Wolf Singer. Role of the temporal domain for response selection and perceptual binding. *Cerebral cortex (New York, NY: 1991)*, 7(6):571–582, 1997.

Andreas K Engel, Pascal Fries, and Wolf Singer. Dynamic predictions: oscillations and synchrony in top–down processing. *Nature Reviews Neuroscience*, 2(10):704–716, 2001.

Martin Engelcke, Adam R Kosiorek, Oiwi Parker Jones, and Ingmar Posner. GENESIS: Generative scene inference and sampling with object-centric latent representations. *International Conference on Learning Represenations (ICLR)*, 2020.

SM Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Geoffrey E Hinton, et al. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, pp. 3225–3233, 2016.

Pascal Fries. A mechanism for cognitive dynamics: neuronal communication through neuronal coherence. *Trends in cognitive sciences*, 9(10):474–480, 2005.

Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms. *International Conference on Learning Represenations (ICLR)*, 2021.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*, pp. 2424–2433, 2019.

Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. On the binding problem in artificial neural networks. *arXiv preprint arXiv:2012.05208*, 2020.

Donald Olding Hebb. *The organisation of behaviour: a neuropsychological theory*. Science Editions New York, 1949.

Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In *International conference on artificial neural networks*, pp. 44–51. Springer, 2011.

Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with em routing. In *International Conference on Learning Representations*, 2018.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.

Laurynas Karazija, Iro Laina, and Christian Rupprecht. Clevrtex: A texture-rich benchmark for unsupervised multi-object segmentation. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Represenations (ICLR)*, 2015.

Thomas Kipf, Gamaleldin F Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff. Conditional object-centric learning from video. *arXiv preprint arXiv:2111.12594*, 2021.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Weihao Sun, Gautam Singh, Fei Deng, Jindong Jiang, and Sungjin Ahn. SPACE: Unsupervised object-oriented scene representation via spatial attention and decomposition. *International Conference on Learning Represenations (ICLR)*, 2020.

Phillip Lippe. UvA Deep Learning Tutorials, 2021. URL `uvadlc-notebooks.readthedocs.io`.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 2020.

Sindy Löwe, Klaus Greff, Rico Jonschkowski, Alexey Dosovitskiy, and Thomas Kipf. Learning object-centric video models by contrasting sets. *arXiv preprint arXiv:2011.10287*, 2020.

Peter M Milner. A model for visual shape recognition. *Psychological review*, 81(6):521, 1974.

Michael C Mozer, Richard S Zemel, Marlene Behrmann, and Christopher KI Williams. Learning to segment images using dynamic feature binding. *Neural Computation*, 4(5):650–665, 1992.

Agostina Palmigiano, Theo Geisel, Fred Wolf, and Demian Battaglia. Flexible information routing by transient synchrony. *Nature neuroscience*, 20(7):1014–1022, 2017.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, 2019.

William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

A Ravishankar Rao and Guillermo A Cecchi. An objective function utilizing complex sparsity for efficient segmentation in multi-layer oscillatory networks. *International Journal of Intelligent Computing and Cybernetics*, 2010.

A Ravishankar Rao and Guillermo A Cecchi. The effects of feedback and lateral connections on perceptual processing: A study using oscillatory networks. In *The 2011 International Joint Conference on Neural Networks*, pp. 1177–1184. IEEE, 2011.

A Ravishankar Rao, Guillermo A Cecchi, Charles C Peck, and James R Kozloski. Unsupervised segmentation with dynamical units. *IEEE transactions on neural networks*, 19(1):168–182, 2008.

Supratim Ray and John HR Maunsell. Differences in gamma frequencies across visual cortex restrict their possible use in computation. *Neuron*, 67(5):885–896, 2010.

David P Reichert and Thomas Serre. Neuronal synchrony in complex-valued deep networks. *International Conference on Learning Represenations (ICLR)*, 2014.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *Advances in Neural Information Processing Systems*, pp. 4967–4976, 2017.

Michael N Shadlen and J Anthony Movshon. Synchrony unbound: review a critical evaluation of the temporal binding hypothesis. *Neuron*, 24:67–77, 1999.

Wolf Singer. Neuronal synchrony: a versatile code for the definition of relations? *Neuron*, 24(1): 49–65, 1999.

Wolf Singer. Distributed processing and temporal codes in neuronal networks. *Cognitive neurodynamics*, 3(3):189–196, 2009.

Wolf Singer and Charles M Gray. Visual feature integration and the temporal correlation hypothesis. *Annual review of neuroscience*, 18(1):555–586, 1995.

Gautam Singh, Fei Deng, and Sungjin Ahn. Illiterate dall-e learns to compose. *arXiv preprint arXiv:2110.11405*, 2021.

Christoph Von Der Malsburg and Werner Schneider. A neural cocktail-party processor. *Biological cybernetics*, 54(1):29–40, 1986.

Richard S Zemel, Christopher KI Williams, and Michael C Mozer. Lending direction to neural networks. *Neural Networks*, 8(4):503–512, 1995.

A Appendix

A.1 Implementation

(Complex) AutoEncoder Table 3 shows the architecture of the Complex AutoEncoder, as well as its real-valued counterpart.

Table 3: Autoencoder architecture used for the Complex AutoEncoder, as well as the real-valued autoencoding baseline.

	Layer	Feature Dimension (H × W × C)	Kernel	Stride	Padding Input / Output	Activation Function
f_{enc}	Conv	$16 \times 16 \times 32$	3	2	1 / 0	(Complex-)ReLU
	Conv	$16 \times 16 \times 32$	3	1	1 / 0	(Complex-)ReLU
	Conv	$8 \times 8 \times 64$	3	2	1 / 0	(Complex-)ReLU
	Conv	$8 \times 8 \times 64$	3	1	1 / 0	(Complex-)ReLU
	Conv	$4 \times 4 \times 64$	3	2	1 / 0	(Complex-)ReLU
	Reshape	$1 \times 1 \times 1024$	-	-	-	-
	Linear	$1 \times 1 \times 64$	-	-	-	(Complex-)ReLU
f_{dec}	Linear	$1 \times 1 \times 1024$	-	-	-	(Complex-)ReLU
	Reshape	$4 \times 4 \times 64$	-	-	-	-
	TransConv	$8 \times 8 \times 64$	3	2	1 / 1	(Complex-)ReLU
	Conv	$8 \times 8 \times 64$	3	1	1 / 0	(Complex-)ReLU
	TransConv	$16 \times 16 \times 32$	3	2	1 / 1	(Complex-)ReLU
	Conv	$16 \times 16 \times 32$	3	1	1 / 0	(Complex-)ReLU
	TransConv	$32 \times 32 \times 1$	3	2	1 / 1	(Complex-)ReLU

We used the default parameter initialization of PyTorch for all layers, except f_{out} for which we set the initial weight $w = 1$ and the initial bias $b = 0$. Additionally, we initialize all phase-biases $\boldsymbol{b}_{\boldsymbol{\varphi}}$ with zero. After the linear layers, we apply Layer Normalization (Ba et al., 2016) instead of Batch Normalization.

We optimized all hyperparameters, except the number of training steps, on the validation set of the 3Shapes dataset and subsequently applied them for the training on all datasets. Across the board, we found that each hyperparameter setting that improved the performance of the Complex AutoEncoder also improved the performance of the real-valued autoencoder and vice versa. As a result, we use the same hyperparameters to train both models.

To create the object-wise reconstructions in Figure 5, we first cluster the features created by the encoder f_{enc} by following the procedure described in Section 3.4. Then, we mask out all values that are not part of a particular cluster with zeros. Finally, we fine-tune the decoder to reconstruct individual objects given these masked out feature vectors for 10,000 steps using Adam with a learning rate of $5\text{e}{-5}$.

SlotAttention To implement SlotAttention, we followed the description and hyperparameters provided by Locatello et al. (2020) as well as their open-source implementation[1]. We used a hidden dimension of 64 throughout the model and adjusted the decoding architecture as described in Table 4 as this improved SlotAttention's performance on our datasets. Besides this final setup that we found to perform best, we tested the following setups on the validation set of the 3Shapes dataset: the decoder setup as used by Locatello et al. (2020) for the Tetrominoes and Multi-dSprites datasets (i.e. spatially broadcast to a resolution of 32×32 and apply four transposed-convolutional layers), a setup in which the fourth transposed-convolutional layer in Table 4 is removed from the decoder, as well as a setup in which the number of channels is halved across all layers. None of these setups learned to disentangle objects on the MNIST&Shape dataset.

Datasets For our experiments, we generate three grayscale datasets: 2Shapes, 3Shapes, and MNIST&Shape. All images within these datasets feature a black background and white objects of differing shapes. In the 2Shapes and 3Shapes datasets, the foreground objects and the background are

[1] `https://github.com/google-research/google-research/tree/master/slot_attention`

Table 4: Architecture used for the Spatial-Broadcast Decoder in the SlotAttention model.

Layer	Feature Dimension (H × W × C)	Kernel	Stride	Padding Input / Output	Activation Function
Spatial Broadcast	$4 \times 4 \times 64$	-	-	-	-
Position Embedding	$4 \times 4 \times 64$	-	-	-	
TransConv	$7 \times 7 \times 64$	5	2	2 / 0	ReLU
TransConv	$15 \times 15 \times 64$	5	2	2 / 0	ReLU
TransConv	$32 \times 32 \times 64$	5	2	2 / 1	ReLU
TransConv	$32 \times 32 \times 64$	3	1	1 / 0	ReLU
TransConv	$32 \times 32 \times 2$	3	1	1 / 0	ReLU

plain white and plain black, respectively, without noise. In the MNIST&Shape dataset, the digits exhibit differing grayscale values. All objects are placed in random locations while ensuring that no part of the object is cut-off at the image boundary.

We use four different object types ($\square, \triangle, \bigtriangledown$, and MNIST digits). The square has an outer side-length of 13 pixels. Both triangles are isosceles triangles, have a base-length of 17 pixels, and are 9 pixels high. Both the square's and the triangles' outlines have a width of 3 pixels.

For the MNIST&Shape dataset, we resize each MNIST digit to match the input image size of our dataset (i.e. 32×32) before applying it to an image. Then, we label pixels as "digit" when their value is > -0.8 after normalization to the $[-1, 1]$ range. This threshold ensures that most of the digit pixels are labeled as such, while minimizing the influence of potentially noisy background pixels. We follow the original dataset split to create the test images and divide the original training set to get 50,000 MNIST digits for our training set and 10,000 MNIST digits for our validation set.

We scale and shift all inputs to the range $[0, 1]$ for the autoencoding models, and we use an input range of $[-1, 1]$ for the SlotAttention model.

A.2 Additional Results

Training Curves In Figure 6, we plot the object discovery performance throughout training of the Complex AutoEncoder and the SlotAttention model on the 2Shapes and 3Shapes datasets. For both datasets, the SlotAttention model keeps improving in performance throughout its 500,000 training steps. The Complex AutoEncoder, on the other hand, converges much faster (within 10,000 – 100,000 steps), leading to significantly faster training times.

Sensitivity Analysis Figure 7 shows the influence of the feature dimension that is output by the CAE's encoder f_{enc} on the model's performance. We find that the model achieves strong performance for a broad range of feature dimensions – indicating that CAE does not require a restricted bottleneck size to create disentangled object representations.

Additional Qualitative Results We highlight the phase separations created by the Complex AutoEncoder in Figure 8, and compare all models on the 2Shapes, 3Shapes and MNIST&Shape datasets in Figures 9, 10 and 11, respectively.

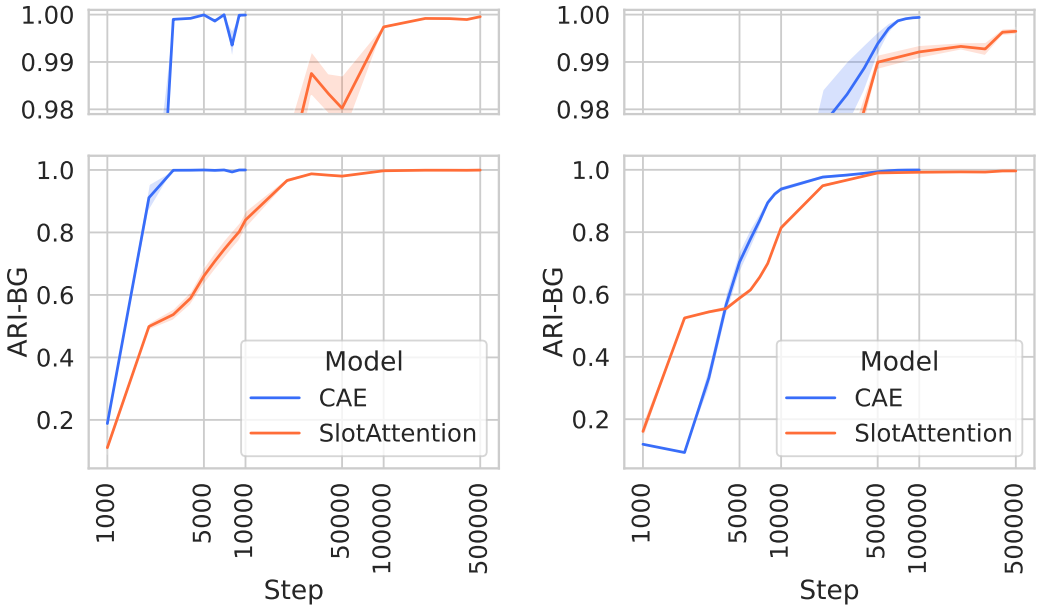

Figure 6: Training Curves. Plotting the ARI-BG scores on the validation set throughout training on the 2Shapes (**Left**) and 3Shapes (**Right**) dataset for the Complex AutoEncoder (CAE) and SlotAttention model (mean $\pm$ standard error across 8 seeds). The CAE achieves comparable or better performance within 5-50 times fewer training steps compared to the SlotAttention model.

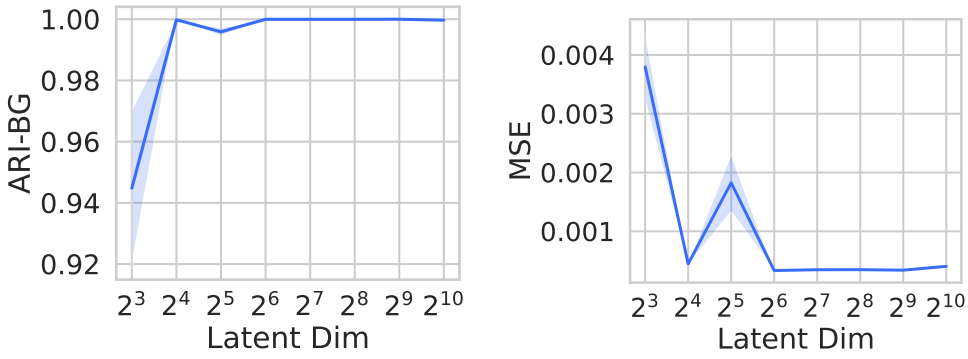

Figure 7: Influence of the latent dimension on CAE's performance on the 2Shapes dataset (mean $\pm$ standard error across 8 seeds). We find that the model does not require a restricted bottleneck to create disentangled object representations.

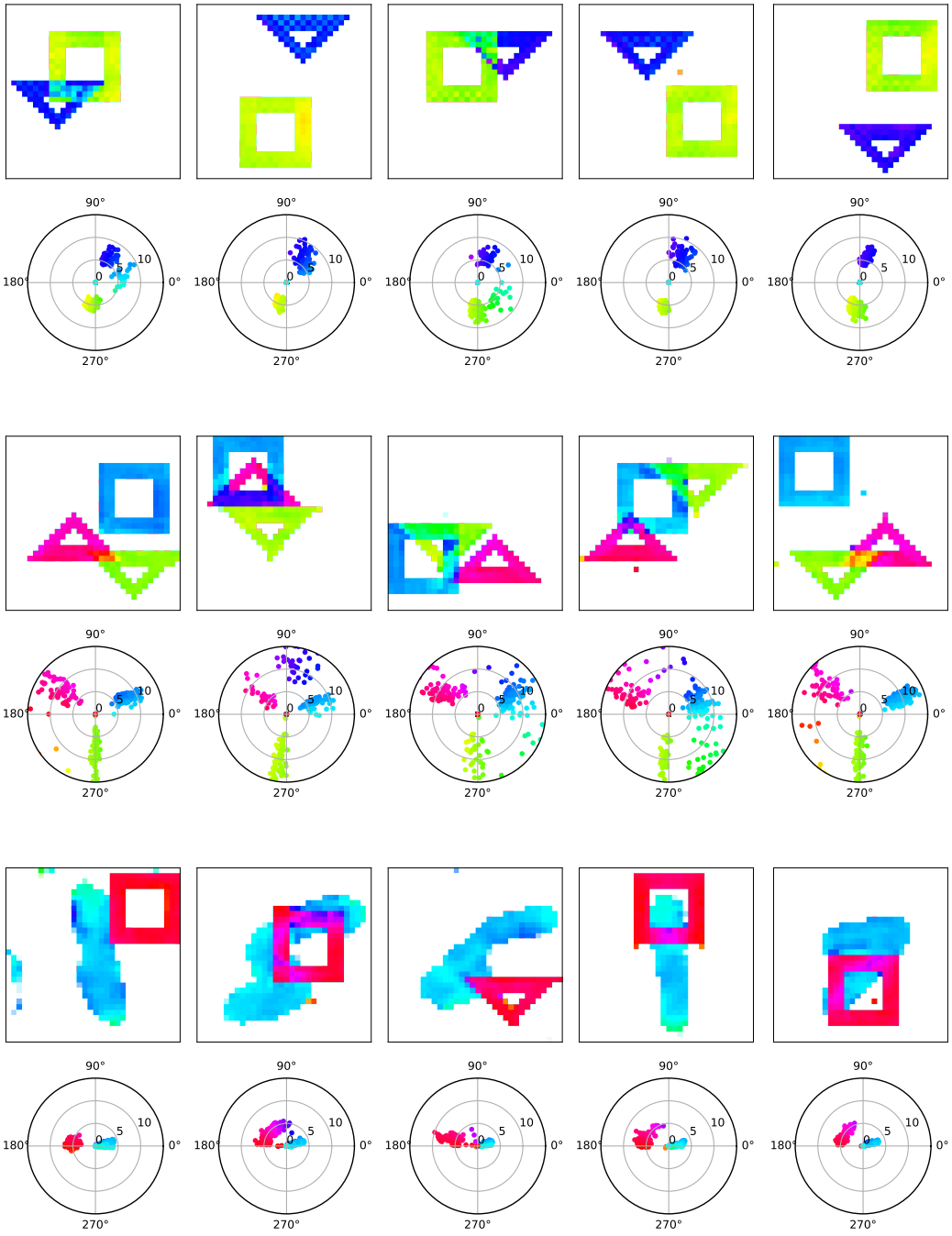

Figure 8: Phase separation in the Complex AutoEncoder on random test-samples from the 2Shapes (**Top**), 3Shapes (**Middle**) and MNIST&Shape datasets (**Bottom**). For each sample, we show the output phase images on top, and the corresponding output values in the complex plane in the bottom (best viewed in color).

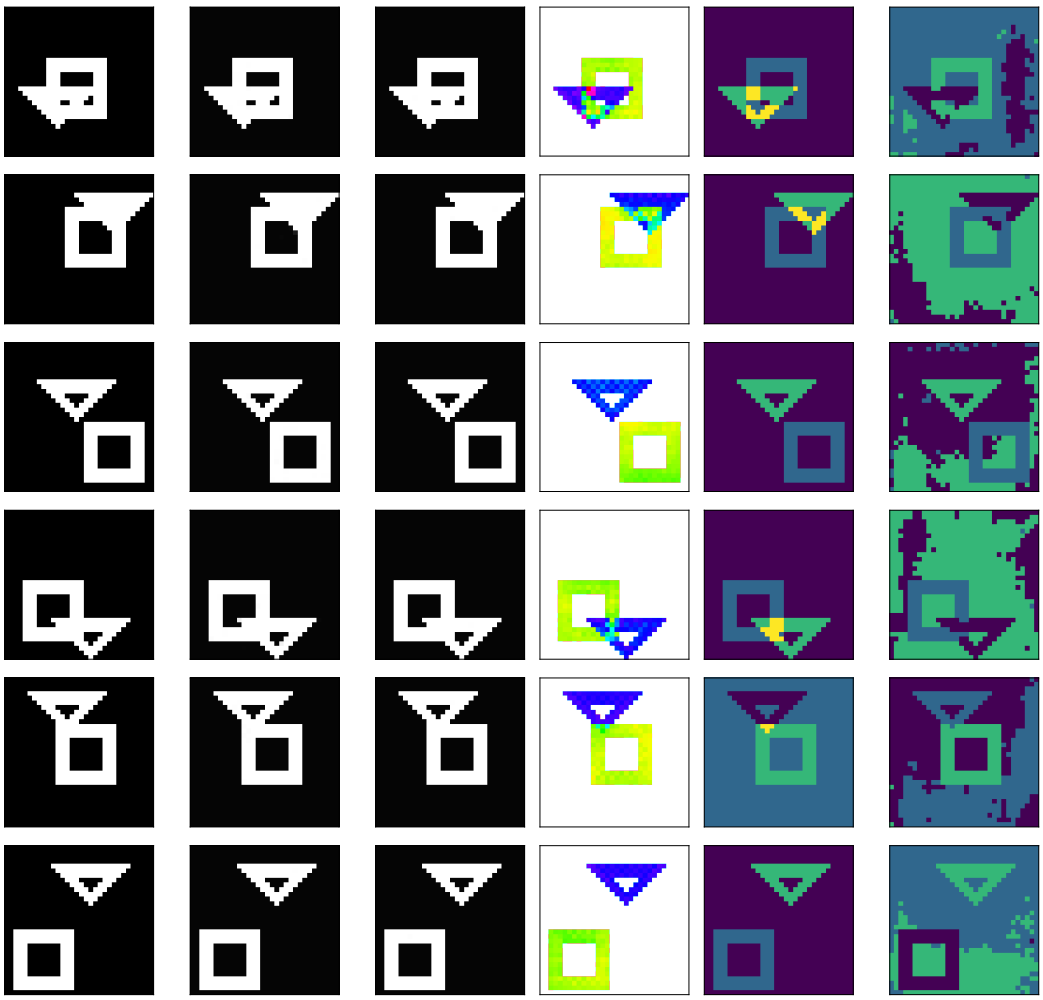

Figure 9: Visual comparison of the performance of the Complex AutoEncoder, its real-valued counterpart (AutoEncoder) and the SlotAttention model on random test-samples from the 2Shapes dataset.

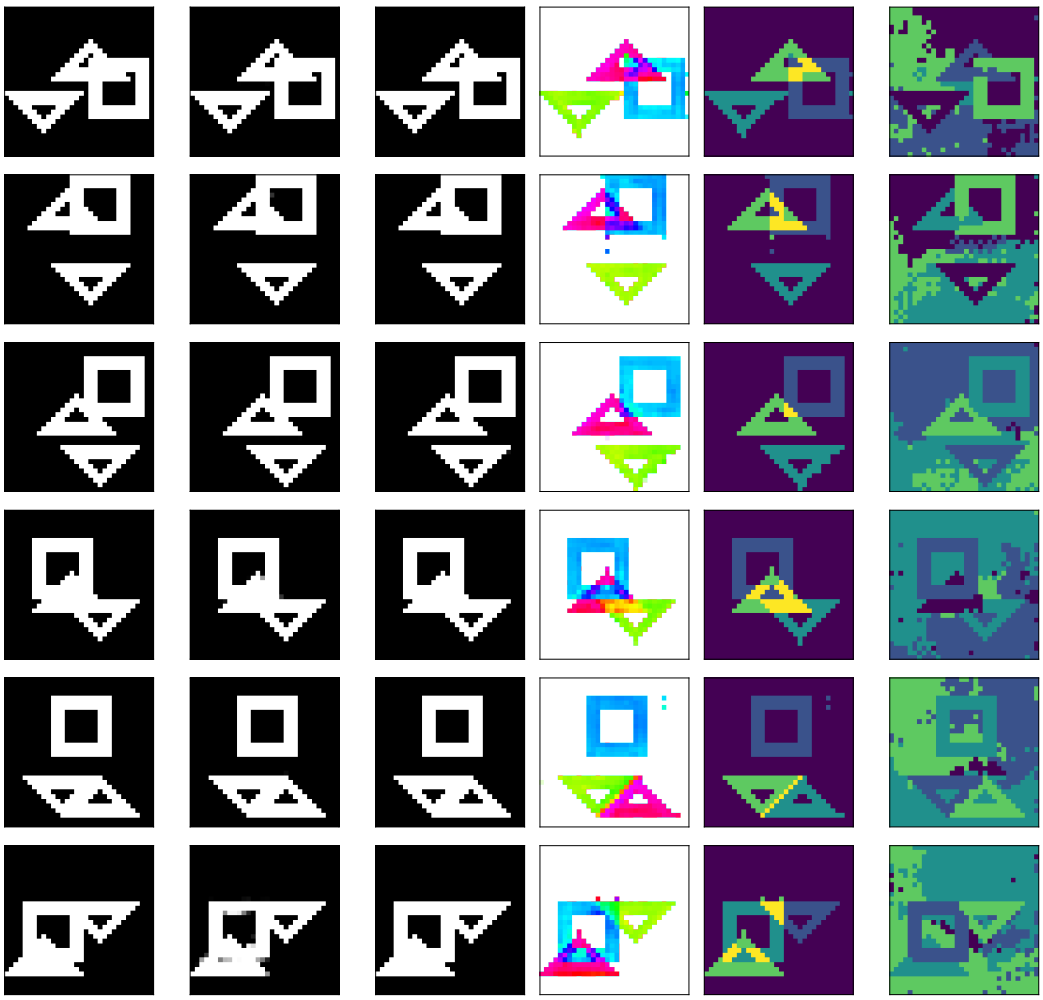

Figure 10: Visual comparison of the performance of the Complex AutoEncoder, its real-valued counterpart (AutoEncoder) and the SlotAttention model on random test-samples from the 3Shapes dataset.

Figure 11: Visual comparison of the performance of the Complex AutoEncoder, its real-valued counterpart (AutoEncoder) and the SlotAttention model on random test-samples from the MNIST&Shape dataset.