ARTIFICIAL KURAMOTO OSCILLATORY NEURONS

Anonymous authors
Paper under double-blind review

ABSTRACT

It has long been known in both neuroscience and AI that "binding" between neurons leads to a form of competitive learning where representations are compressed in order to represent more abstract concepts in deeper layers of the network. More recently, it was also hypothesized that dynamic (spatiotemporal) representations play an important role in both neuroscience and AI. Building on these ideas, we introduce Artificial Kuramoto Oscillatory Neurons (*AKOrN*) as a dynamical alternative to threshold units, which can be combined with arbitrary connectivity designs such as fully connected, convolutional, or attentive mechanisms. Our generalized Kuramoto updates bind neurons together through their synchronization dynamics. We show that this idea provides performance improvements across a wide spectrum of tasks such as unsupervised object discovery, adversarial robustness, calibrated uncertainty quantification, and reasoning. We believe that these empirical results show the importance of rethinking our assumptions at the most basic neuronal level of neural representation, and in particular show the importance of dynamical representations

1 INTRODUCTION

Before the advent of modern deep learning architectures, artificial neural networks were inspired by biological neurons. In contrast to the McCulloch-Pitts neuron (McCulloch & Pitts, 1943) which was designed as an abstraction of an integrate-and-fire neuron (Sherrington, 1906), recent building blocks of neural networks are designed to work well on modern hardware (Hooker, 2021). As our understanding of the brain is improving over recent years, and neuroscientists are discovering more about its information processing principles, we can ask ourselves again if there are lessons from neuroscience that can be used as design principles for artificial neural nets.

In this paper, we follow a more modern dynamical view of neurons as oscillatory units that are coupled to other neurons (Muller et al., 2018). Similar to how the binary state of a McCulloch-Pitts neuron abstracts the firing of a real neuron, we will abstract an oscillating neuron by an N-dimensional unit vector that rotates on the sphere (Löwe et al., 2023). We build a new neural network architecture that has iterative modules that update N-dimensional oscillatory neurons via a generalization of the well-known non-linear dynamical model called the Kuramoto model (Kuramoto, 1984).

The Kuramoto model describes the synchronization of oscillators; each Kuramoto update applies forces to connected oscillators, encouraging them to become aligned or anti-aligned. This process is similar to binding in neuroscience and can be understood as distributed and continuous clustering. Thus, networks with this mechanism tend to compress their representations via synchronization.

We incorporate the Kuramoto model into an artificial neural network, by applying the differential equation that describes the Kuramoto model to each individual neuron. The resulting artificial Kuramoto oscillatory neurons (*AKOrN*) can be combined with layer architectures such as fully connected layers, convolutions, and attention mechanisms.

We explore the capabilities of *AKOrN* and find that its neuronal mechanism drastically changes the behavior of the network. *AKOrN* strongly binds object features with competitive performance to slot-based models in object discovery, enhances the reasoning capability of self-attention, and increases robustness against random, adversarial, and natural perturbations with surprisingly good calibration.

2 MOTIVATION

It was recognized early on that neurons interact via lateral connections (Hubel & Wiesel, 1962; Somers et al., 1995). In fact, neighboring neurons tend to cluster their activities (Gray et al., 1989; Mountcastle, 1997), and clusters tend to compete to explain the input. This “competitive learning” has the advantage that information is compressed as we move through the layers, facilitating the process of abstraction by creating an information bottleneck (Amari & Arbib, 1977). Additionally, the competition encourages different higher-level neurons to focus on different aspects of the input (i.e. they specialize). This process is made possible by synchronization: like fireflies in the night, neurons tend to synchronize their activities with their neighbors’, which leads to the compression of their representations. This idea has been used in artificial neural networks before to model “binding” between neurons, where neurons representing features such as square, blue, and toy are bound by synchronization to represent a square blue toy (Reichert & Serre, 2013; Löwe et al., 2022). In this paper, we will use an N-dimensional generalization of the famous Kuramoto model (Kuramoto, 1984) to model this synchronization.

Our model has the advantage that it naturally incorporates spatiotemporal representations in the form of traveling waves (Keller et al., 2024), for which there is ample evidence in the neuroscientific literature. While their role in the brain remains poorly understood, it has been postulated that they are involved in short-term memory, long-range coordination between brain regions, and other cognitive functions (Rubino et al., 2006; Lubenov & Siapas, 2009; Fell & Axmacher, 2011; Zhang et al., 2018; Roberts et al., 2019; Muller et al., 2016; Davis et al., 2020; Benigno et al., 2023). For example, Muller et al. (2016) finds that oscillatory patterns in the thalamocortical network during sleep are organized into circular wave-like patterns, which could give an account of how memories are consolidated in the brain. Davis et al. (2020) suggest that spontaneous traveling waves in the visual cortex modulate synaptic activities and thus act as a gating mechanism in the brain. In the generalized Kuramoto model, traveling waves naturally emerge as neighboring oscillators start to synchronize (see on the left in Fig. 1, and Fig. 10 in the Appendix).

Another advantage of using dynamical neurons is that they can perform a form of reasoning. Kuramoto oscillators have been successfully used to solve combinatorial optimization tasks such as k-SAT problems (Heisenberg, 1985; Wang & Roychowdhury, 2017). This can be understood by the fact that Kuramoto models can be viewed as continuous versions of discrete Ising models, where phase variables replace the discrete spin states. Many authors have argued that the modern architectures based on, e.g., transformers lack this intrinsic capability of “neuro-symbolic reasoning” (Dziri et al., 2024; Bounsi et al., 2024). We show that *AKOrN* can successfully solve Sudoku puzzles, illustrating this capability. Additionally, *AKOrN* relates to models in quantum physics and active matter (see appendix B).

In summary, *AKOrN* combines beneficial features such as competitive learning (i.e., feature binding), reasoning, robustness and uncertainty quantification, as well as the potential advantages of traveling waves observed in the brain, while being firmly grounded in well-understood physics models.

3 THE KURAMOTO MODEL

The Kuramoto model (Kuramoto, 1984) is a non-linear dynamical model of oscillators, that exhibits synchronization phenomena. Even with its simple formulation, the model can represent numerous dynamical patterns depending on the connections between oscillators (Breakspear et al., 2010; Heitmann et al., 2012).

In the original Kuramoto model, each oscillator i is represented by its phase information $\theta_i \in [0, 2\pi)$. The differential equation of the Kuramoto model is

$$\dot{\theta}_i = \omega_i + \textstyle\sum_j J_{ij} \sin(\theta_j - \theta_i), \tag{1}$$

where $\omega_i \in \mathbb{R}$ is the natural frequency and $J_{ij} \in \mathbb{R}$ represents the connections between oscillators: if $J_{ij} > 0$ the i and j-th oscillator tend to align, and if $J_{ij} < 0$, they tend to oppose each other.

While the original Kuramoto model describes one-dimensional oscillators, we use a *multi-dimensional vector version* of the model (Olfati-Saber, 2006; Zhu, 2013; Chandra et al., 2019; Lipton et al., 2021) with a symmetry-breaking term into neural networks. We denote oscillators by

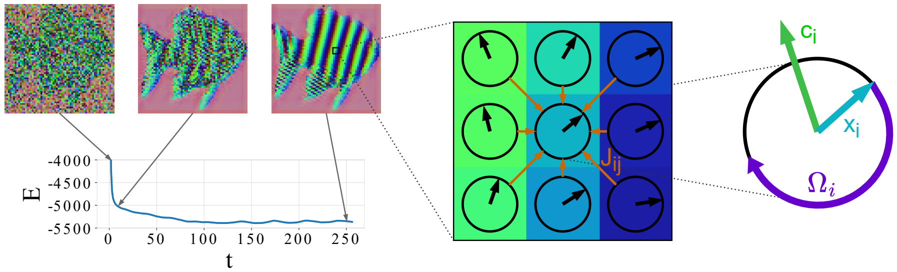


Figure 1: Our proposed artificial Kuramoto oscillatory neurons (*AKOrN*). The series of pictures on the left are 64× 64 Kuramoto oscillators evolving by the Kuramoto updates (Eq. (2)), along with a plot of the energies computed by Eq. (3). Each single oscillator $\mathbf{x}_i$ is an N-dimensional vector on the sphere and is influenced by (1) connected oscillators through the weights $\mathbf{J}_{ij}$, (2) conditional stimuli $\mathbf{c}_i$, and (3) $\mathbf{\Omega}_i$ that determines the natural frequency of each oscillator. See Fig. 10 for details on $\mathbf{C}$ and $\mathbf{J}$.

$\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{C}$, where each $\mathbf{x}_i$ is a vector on a hypersphere: $\mathbf{x}_i \in \mathbb{R}^N$, $\|\mathbf{x}_i\|_2 = 1$. N is each single oscillator dimension called *rotating dimensions* and C is the number of oscillators. While each $\mathbf{x}_i$ is time-dependent, we omit t for clarity. The oscillator index i may have multiple dimensions: if the input is an image, for example, each oscillator is represented by $\mathbf{x}_{c,h,w}$ with c, h, w indicating channel, height and width positions, respectively.

The differential equation of our vector-valued Kuramoto model is written as follows:

$$\dot{\mathbf{x}}_i = \mathbf{\Omega}_i \mathbf{x}_i + \mathrm{Proj}_{\mathbf{x}_i}(\mathbf{c}_i + \sum_j \mathbf{J}_{ij}\mathbf{x}_j) \text{ where } \mathrm{Proj}_{\mathbf{x}_i}(\mathbf{y}_i) = \mathbf{y}_i - \langle \mathbf{y}_i, \mathbf{x}_i \rangle \, \mathbf{x}_i \tag{2}$$

Here, $\mathbf{\Omega}_i$ is an $N \times N$ anti-symmetric matrix and $\mathbf{\Omega}_i \mathbf{x}_i$ is called the natural frequency term that determines each oscillator's own rotation frequency and angle. The second term governs interactions between oscillators, where $\mathrm{Proj}_{\mathbf{x}_i}$ is an operator that projects an input vector onto the tangent space of the sphere at $\mathbf{x}_i$. We show a visual description of $\mathrm{Proj}_{\mathbf{x}_i}$ and a relation between the vector valued Kuramoto model and the original one in the Appendix A.1. $\mathbf{C} = \{\mathbf{c}_i\}_{i=1}^{C}, \mathbf{c}_i \in \mathbb{R}^N$ is a data-dependent variable, which is computed from the observational input or the activations of the previous layer. In this paper, every $\mathbf{c}_i$ is set to be constant across time, but it can be a time-dependent variable. $\mathbf{c}_i$ can be seen as another oscillator that has a unidirectional connection to $\mathbf{x}_i$. Since $\mathbf{c}_i$ is not affected by any oscillators, $\mathbf{c}_i$ strongly binds $\mathbf{x}_i$ to the same direction as $\mathbf{c}_i$, i.e. it acts as a bias direction (see Fig. 10 in the Appendix). In physics lingo, $\mathbf{C}$ is often referred to as a "symmetry breaking" field.

The Kuramoto model is Lyapunov if we assume certain symmetric properties in $\mathbf{J}_{ij}$ and $\mathbf{\Omega}_i$ (Aoyagi, 1995; Wang & Roychowdhury, 2017). For example, if $\mathbf{J}$ is symmetric and different oscillators share the same natural frequencies: $\mathbf{J}_{ij} = \mathbf{J}_{ji}^{\mathrm{T}}$, $\mathbf{\Omega}_i = \mathbf{\Omega}$, and $\mathbf{\Omega}\mathbf{c}_i = \mathbf{0}$, each update is guaranteed to minimize the following energy:

$$E = -\sum_{i,j} \mathbf{x}_i^{\mathrm{T}} \mathbf{J}_{ij} \mathbf{x}_j - \sum_i \mathbf{c}_i^{\mathrm{T}} \mathbf{x}_i \tag{3}$$

Fig. 1 on the left shows how the Kuramoto oscillators and the corresponding energy evolve with a simple Gaussian kernel as the connectivity matrix. Here, we set $\mathbf{C}$ as a silhouette of a fish, where $\mathbf{c}_i = \mathbf{1}$ on the outer silhouette and $\mathbf{c}_i = \mathbf{0}$ on the inner silhouette. The oscillator state is initially disordered, but gradually exhibits collective behavior, eventually becoming a spatially propagating wavy pattern. We include animations of visualized oscillators, including oscillators of trained *AKOrN* models used in our experiments, in the Supplementary Material.

We would like to note that we found that even without symmetric constraints, the energy value decreases relatively stably, and the models perform better across all tasks we tested compared to models with symmetric $\mathbf{J}$. A similar observation is made by Effenberger et al. (2022) where heterogeneous oscillators such as those with different natural frequencies are helpful for the network to control the level of synchronization and increase the network capacity. From here, we assume no symmetric constraints on $\mathbf{J}$ and $\mathbf{\Omega}$. Having asymmetric (a.k.a. non-reciprocal) connections is aligned with the biological neurons in the brain, which also do not have symmetric synapses.

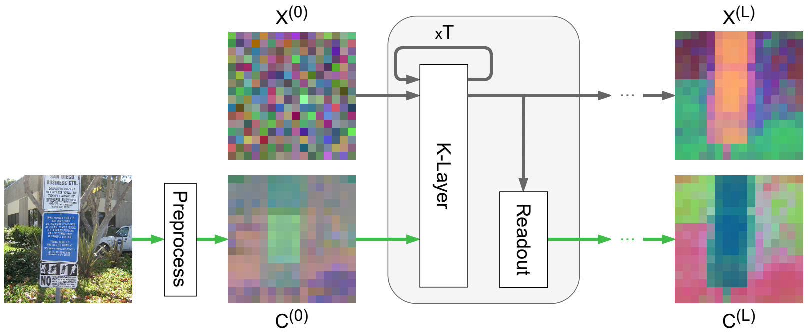


Figure 2: Our proposed Kuramoto-based network (here, for image processing). Each block consists of a Kuramoto-layer and a readout module described in Sec 4. $\mathbf{C}^{(L)}$ is used to make the final prediction of our model.

4 NETWORKS WITH KURAMOTO OSCILLATORS

We utilize the artificial Kuramoto oscillator neurons (*AKOrN*) as a basic unit of information processing in neural networks (Fig. 2). First, we transform an observation with a relatively simple function to create the initial conditional stimuli $\mathbf{C}^{(0)}$. Next, $\mathbf{X}^{(0)}$ is initialized by either $\mathbf{C}^{(0)}$, a fixed learned embedding, random vectors, or a mixture of these initialization schemes. The block is composed of two modules: the Kuramoto layer and the readout module, which together process the pair $\{\mathbf{X}, \mathbf{C}\}$. The Kuramoto layer updates $\mathbf{X}$ with the conditional stimuli $\mathbf{C}$, and the readout layer extracts features from the final oscillatory states to create new conditional stimuli. We denote the number of layers by L, and l-th layer's output by $\{\mathbf{X}^{(l)}, \mathbf{C}^{(l)}\}$.

Kuramoto layer Starting with $\mathbf{X}^{(l,0)} := \mathbf{X}^{(l-1)}$ as initial oscillators, where the second superscript denotes the time step, we update them by the discrete version of the differential equation (2):

$$\Delta \mathbf{x}_i^{(l,t)} = \boldsymbol{\Omega}_i^{(l)} \mathbf{x}_i^{(l,t)} + \mathrm{Proj}_{\mathbf{x}_i^{(l,t)}}(\mathbf{c}_i^{(l-1)} + \sum_j \mathbf{J}_{ij}^{(l)} \mathbf{x}_j^{(l,t)}) \tag{4}$$

$$\mathbf{x}_i^{(l,t+1)} = \Pi\left[\mathbf{x}_i^{(l,t)} + \gamma \Delta \mathbf{x}_i^{(l,t)}\right], \tag{5}$$

where Π is the normalizing operator $\mathbf{x}/\|\mathbf{x}\|_2$ that ensures that the oscillators stay on the sphere. $\gamma > 0$ is a scalar controlling the step size of the update, which is learned in our experiments. We call this update a Kuramoto update or a Kuramoto step from here. We optimize both $\boldsymbol{\Omega}^{(l)}$ and $\mathbf{J}^{(l)}$ given the task objective.

We update the oscillators T times. We denote the oscillators at T by $\mathbf{X}^{(l,T)}$. This oscillator state is used as the initial state of the next block: $\mathbf{X}^{(l)} := \mathbf{X}^{(l,T)}$.

Readout module We read out patterns encoded in the oscillators to create new conditional stimuli $\mathbf{C}^{(l)}$ for the subsequent block. Since the oscillators are constrained onto the (unit) hyper-sphere, all the information is encoded in their directions. In particular, the relative direction between oscillators is an important source of information because patterns after certain Kuramoto steps only differ in global phase shifts (see the last two patterns in Fig. 10 in the Appendix). To capture phase invariant patterns, we take the norm of the linearly processed oscillators:

$$\mathbf{C}^{(l)} = g(\mathbf{m}) \in \mathbb{R}^{C' \times N}, m_k = \|\mathbf{z}_k\|_2,\ \mathbf{z}_k = \textstyle\sum_i \mathbf{U}_{ki} \mathbf{x}_i^{(l,T)} \in \mathbb{R}^{N'}, \tag{6}$$

where $\mathbf{U}_{ki} \in \mathbb{R}^{N' \times N}$ is a learned weight matrix, g is a learned function, and $\mathbf{m} = [m_1, ..., m_K]^{\mathrm{T}} \in \mathbb{R}^K$. N' is typically set to the same value as N. In this work, g is just the identity function, a linear layer, or at most a three-layer neural network with residual connections. Because the module computes the norm of (weighted) $\mathbf{X}^{(l,T)}$, this readout module includes functions that are invariant to the global phase shift in the solution space. Unless otherwise specified, we set $C' = C$ and $K = C \times N$ in all our experiments.

4.1 CONNECTIVITIES

We implement artificial Kuramoto oscillator neurons (*AKOrN*) within convolutional and self-attention layers. We write down the formal equations of the connectivity for completeness, however, they simply follow the conventional operation of convolution or self-attention applied to oscillatory neurons flattened w.r.t the rotating dimension N. In short, convolutional connectivity is local, and attentive connectivity is dynamic input-dependent connectivity.

Convolutional connectivity To implement *AKOrN* in a convolutional layer, oscillators and conditional stimuli are represented as $\{\mathbf{x}_{c,h,w}, \mathbf{c}_{c,h,w}\}$ where c, h, w are channel, height and width positions, and the update direction is given by:

$$\mathbf{y}_{c,h,w} := \mathbf{c}_{c,h,w} + \sum_{d} \sum_{h',w' \in R[H',W']} \mathbf{J}_{c,d,h',w'} \mathbf{x}_{d,(h+h'),(w+w')}, \tag{7}$$

where $R[H', W'] = [1, ..., H'] \times [1, ..., W']$ is the $H' \times W'$ rectangle region (i.e. kernel size) and $\mathbf{J}_{c,d,h',w'} \in \mathbb{R}^{N \times N}$ are the learned weights in the convolution kernel where $(c, d), (h', w')$ are output and input channels, and height and width positions.

Attentive connectivity Similar to Bahdanau et al. (2014); Vaswani et al. (2017), we construct the internal connectivity in the QKV-attention manner. In this case, oscillators and conditional stimuli are represented by $\{\mathbf{x}_{l,i}, \mathbf{c}_{l,i}\}$ where l and i are indices of tokens and channels, respectively. The update direction becomes:

$$\mathbf{y}_{l,i} := \mathbf{c}_{l,i} + \sum_{m,j} \mathbf{J}_{l,m,i,j} \mathbf{x}_{m,j} = \mathbf{c}_{l,i} + \sum_{m,j} \sum_{k,h} \mathbf{W}^{O}_{h,i,k} A_h(l,m) \mathbf{W}^{V}_{h,k,j} \mathbf{x}_{m,j} \tag{8}$$

$$A_h(l,m) = \frac{e^{d_h(l,m)}}{\sum_m e^{d_h(l,m)}}, \; d_h(l,m) = \sum_a \left\langle \sum_i \mathbf{W}^{Q}_{h,a,i} \mathbf{x}_{l,i}, \sum_i \mathbf{W}^{K}_{h,a,i} \mathbf{x}_{m,i} \right\rangle \tag{9}$$

where $\mathbf{W}^{O}_{h,i,k}, \mathbf{W}^{V}_{h,k,j}, \mathbf{W}^{Q}_{h,a,i}, \mathbf{W}^{K}_{h,a,i} \in \mathbb{R}^{N \times N}$ are learned weights of head h. Since the connectivity is dependent on the oscillator values and thus not static during the updates, it is unclear whether the energy defined in Eq, (3) is proper. Nonetheless, in our experiments, the energy and oscillator states are stable after several updates (see the Supplementary Material, which includes visualizations of the oscillators of trained *AKOrN* models and their corresponding energies over timesteps).

5 RELATED WORKS

The Kuramoto model is rarely seen in machine learning, especially in deep learning. However, several works motivate us to use the Kuramoto model as a mechanism for learning binding features. For example, although tested only in fairly synthetic settings, Liboni et al. (2023) show that cluster features emerge in the oscillators of the Kuramoto model with lateral connections without optimization. Ricci et al. (2021) studies how data-dependent connectivity can construct synchrony on synthetic examples. Also, a line of works on neural synchrony (Reichert & Serre, 2013; Löwe et al., 2022; Stanić et al., 2023; Zheng et al., 2023; Löwe et al., 2023; Gopalakrishnan et al., 2024) shares the same philosophy with *AKOrN*. Zheng et al. (2023) model synchrony by using temporal spiking neurons based on biological neuronal mechanisms. Löwe et al. (2023) extend the concept of complex-valued neurons—used by Reichert & Serre (2013); Löwe et al. (2022) to abstract temporal neurons—into multidimensional neurons. They show that, together with a specific activation function called χ-*binding* that implements the 'winner-take-all' mechanism at the single neuron level (Löwe et al., 2024), the multidimensional neurons learn to encode binding information in their orientations. Those synchrony-based models are shown to work well on relatively synthetic data but have been struggling to scale to natural images. Löwe et al. (2023) shows that their model can work with a large pre-trained self-supervised learning (SSL) model as a feature extractor, but its performance improvement is limited compared to slot-based models.

Slot-based models (Le Roux et al., 2011; Burgess et al., 2019; Greff et al., 2019; Locatello et al., 2020) are the most-used model for object-centric (OC) learning. Their discrete nature of representations is shown to be a good inductive bias to learn such OC representations. However, similarly to synchrony-based models, these models struggle on natural images and are therefore often combined

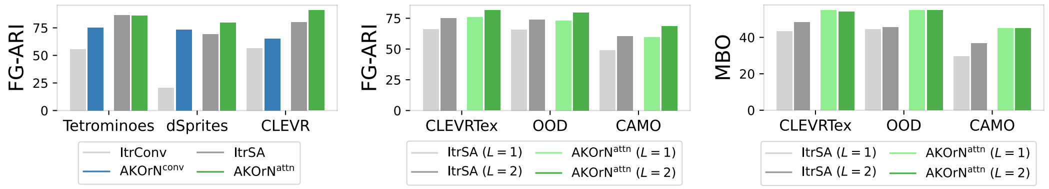


Figure 3: Object discovery performance on synthetic datasets.

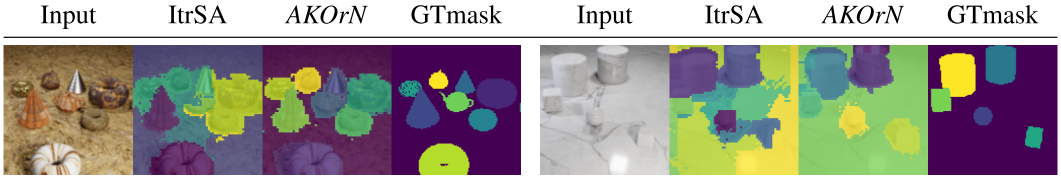


Figure 4: *AKOrN* learns more object-bound features than the non-Kuramoto model counterpart.

Model	CLEVRTex FG-ARI	CLEVRTex MBO	OOD FG-ARI	OOD MBO	CAMO FG-ARI	CAMO MBO
*MONet (Burgess et al., 2019)	19.78	-	37.29	-	31.52	-
SLATE (Singh et al., 2021)	44.19	50.88	-	-	-	-
*Slot-Attention (Locatello et al., 2020)	62.40	-	58.45	-	57.54	-
Slot-diffusion (Wu et al., 2023)	69.66	61.94	-	-	-	-
Slot-diffusion+BO (Wu et al., 2023)	78.50	68.68	-	-	-	-
*DTI (Monnier et al., 2021)	79.90	-	73.67	-	72.90	-
*I-SA (Chang et al., 2022)	78.96	-	83.71	-	57.20	-
BO-SA (Jia et al., 2023)	80.47	-	86.50	-	63.71	-
ISA-TS (Biza et al., 2023)	92.9	-	84.4	-	86.2	-
AKOrN$^{\text{attn}}$	89.24	60.02	88.00	60.96	77.18	53.43

Table 1: Object discovery performance on CLEVRTex and its variants (OOD, CAMO). *AKOrN* is compared among models trained from scratch. *Numbers taken from Jia et al. (2023).

with powerful, pre-trained SSL models such as DINO (Caron et al., 2021). Our proposed continuous Kuramoto neurons can be a building block of the SSL network itself, and we show that they learn better object-centric features than well-known SSL models. Our work is the first work that demonstrates that a synchrony-based model is solely scaled up to natural images.

*AKOrN*s perform particularly well on object discovery tasks when implemented in self-attention layers. Self-attention updates with normalization have been shown mathematically to cluster token features (Geshkovski et al., 2024). Our work combines this clustering behavior of transformers with the clustering induced by the synchronization of the Kuramoto neurons, resulting in *AKOrN* being the first competitive method to slot-based approaches.

Finally, there exist several works on interpreting self-attention in the context of the Hopfield networks (Ramsauer et al., 2020; Hoover et al., 2023). Energy transformer (Hoover et al., 2023) introduces a symmetrized attention mechanism to guarantee the update minimizing certain energy. However, we find such symmetric models worsen the performance in our reasoning task. Our Kuramoto-based models differ from these approaches: the use of unit-norm-constrained neurons with asymmetric connections in $\mathbf{J}$, and their symmetry-breaking term $\mathbf{C}$. These elements contribute to performance improvement over the approach by (Hoover et al., 2023) and conventional self-attention in the reasoning task of our experiments.

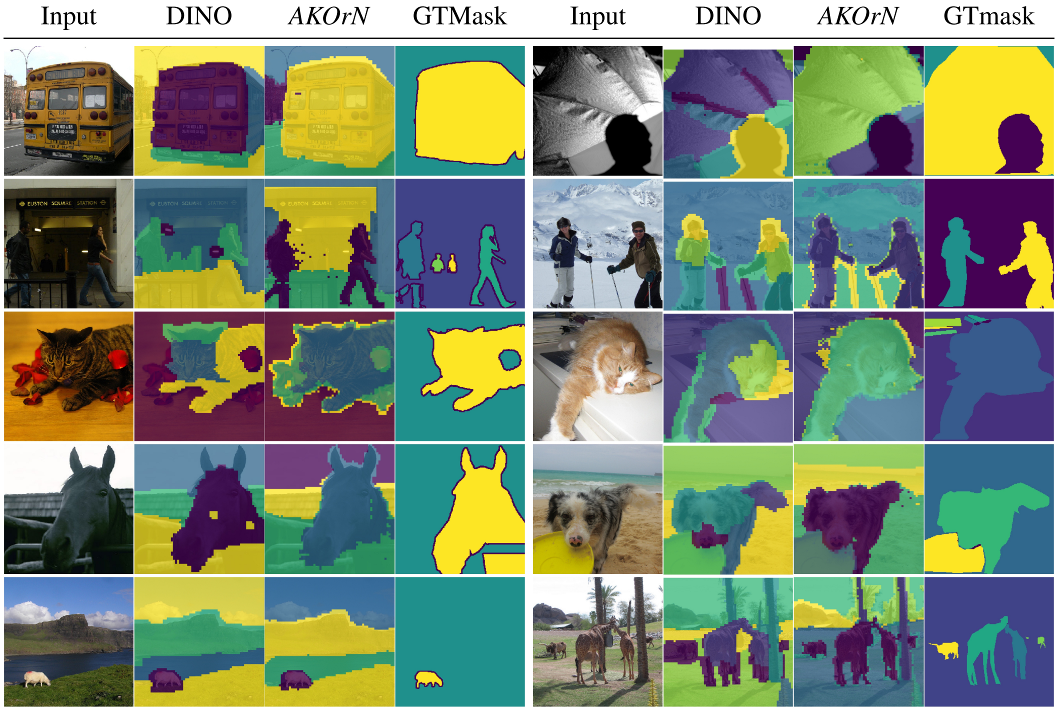


Figure 5: Visualization of clusters on (Left) PascalVOC and (Right) COCO2017.

6 EXPERIMENTS

6.1 UNSUPERVISED OBJECT DISCOVERY

Unsupervised object discovery is the task of finding objects in an image without supervision. Here, we test *AKOrN* on four synthetic datasets (Tetrominoes, dSprites, CLEVR (Kabra et al., 2019), CLEVRTex (Karazija et al., 2021)) and two real image datasets (PascalVOC (Everingham et al., 2010), COCO2017 (Lin et al., 2014)) (see the Appendix D for details). Among the four synthetic datasets, CLEVRTex has the most complex objects and backgrounds. We further evaluate the models trained on the CLEVRTex dataset on two variants (OOD, CAMO). The materials and shapes of objects in OOD differ from those in CLEVRTex, while CAMO (short for camouflage) features scenes where objects and backgrounds share similar textures within each scene.

As baselines, we train models that are similar to ResNet (He et al., 2016) and ViT (Dosovitskiy et al., 2021), but iterate the convolution or self-attention layers multiple times with shared parameters. This allows us to evaluate the impact of our proposed, Kuramoto-based iterative updates. We denote these baselines as Iterative Convolution (ItrConv) and Iterative Self-Attention (ItrSA), respectively. Fig. 11 in the Appendix shows diagrams of each network.

In *AKOrN*, $\mathbf{C}$ is initialized by the patched features of the images, while each $\mathbf{x}_i$ is initialized by random oscillators sampled from the uniform distribution on the sphere. We train the *AKOrN* model with the self-supervised SimCLR (Chen et al., 2020) objective.

We train each model from scratch on the four synthetic datasets. For the two real image datasets, we first train *AKOrN* on ImageNet (Krizhevsky et al., 2012) and directly evaluate that ImageNet-pretrained model on both datasets without fine-tuning. When evaluating, we apply clustering to the final block's output features (In *AKOrN*, it is $\mathbf{C}^{(L)}$). We use agglomeration clustering with average linkage, which we found to outperform K-means for both the baseline models and *AKOrN*. We evaluate the clustering results by foreground adjusted rand index (FG-ARI) and Mean-Best-Overlap (MBO). FG-ARI measures the similarity between the ground truth masks and the computed clusters, only for foreground objects. MBO first assigns each cluster to the highest overlapping ground truth mask and then computes the average intersection-over-union (IoU) of all pairs. See D.1.1 for details.

Model	PascalVOC MBO_i	PascalVOC MBO_c	COCO2017 MBO_i	COCO2017 MBO_c
(slot-based models)				
Slot-attention (Locatello et al., 2020)	22.2	23.7	24.6	24.9
SLATE (Singh et al., 2021)	35.9	41.5	29.1	33.6
(DINO + slot-based model)				
DINOSAUR (Seitzer et al., 2023)	44.0	51.2	31.6	39.7
Slot-diffusion (Wu et al., 2023)	50.4	55.3	31.0	35.0
SPOT (Kakogeorgiou et al., 2024)	48.3	55.6	**35.0**	**44.7**
(transformer + SSL)				
MAE (He et al., 2022)	34.0	38.3	23.1	28.5
MoCoV3 (Chen et al., 2021)	47.3	53.0	28.7	36.0
DINO (Caron et al., 2021)	47.2	53.5	29.4	37.0
AKOrN	**52.0**	**60.3**	31.3	40.3

Table 2: Object discovery on PascalVOC and COCO2017.

For PascalVOC and COCO2017, we show instance-level MBO (MBO_i) and class-level (MBO_c) segmentation results.

***AKOrN* binds object features** Fig. 3 shows that *AKOrN*s improve the object discovery performance over their non-Kuramoto counterparts in almost every dataset (except for Tetrominoes). Interestingly, we observe that convolution is less effective than attention in most datasets. In Fig. 4, we see that the Kuramoto models' clusters are well-aligned with the individual objects, while clusters of the ItrSA model often span across objects and background, and are sensitive to the texture of the background (more clustering results are shown in Figs 30-32 in the Appendix).

Tab. 1 shows a comparison to existing works on CLEVRTex and its variants. All other methods are slot-based. Among the distributed representation models, *AKOrN is the first method that is shown to be competitive with slot-based models on the complex CLEVRTex dataset*.

***AKOrN* scales to natural images** Fig. 5 shows *AKOrN* binds object features on natural images much better than DINO (Caron et al., 2021). We show a benchmark comparison on Pascal VOC and COCO2017 in Tab. 2. The proposed AKOrN model outperforms existing SSL models including DINO, MoCoV3, and MAE on both datasets, showing that it learns more object-bound features than conventional transformer-based models. On Pascal, *AKOrN* is considerably better than other models including models trained from scratch and models trained on features of a pretrained DINO model. On COCO, *AKOrN* again outperforms methods that are trained from scratch and is competitive to DINOSAUR and Slot-diffusion, but is outperformed by the recent SPOT model.

6.2 SOLVING SUDOKU

To test *AKOrN*'s reasoning capability, we apply it on the Sudoku puzzle datasets (Wang et al., 2019; Palm et al., 2018). The training set contains boards with 31-42 given digits. We test models in-distribution (ID) and out-of-distribution (OOD) scenarios. The ID test set contains 1,000 boards sampled from the same distribution, while boards in the OOD set contain much fewer given digits (17-34) than the train set. To initialize $\mathbf{C}$, we use embeddings of the digits 0-9 (0 for blank, 1-9 for given digits). The initial $\mathbf{x}_i$ takes the value $\mathbf{c}_i/\|\mathbf{c}_i\|_2$ when a digit is given, and is randomly sampled from the uniform distribution on the sphere for blank squares. The number of Kuramoto steps during training is set to 16. We also train a transformer model with 8 blocks.

***AKOrN* solves Sudoku puzzles** *AKOrN* perfectly solves all puzzles in the ID test set, while only Recurrent Transformer (R-Transformer (Yang et al., 2023)) achieves this (Tab. 3). On the OOD set, *AKOrN* achieves 61.1±14.7 accuracy which is on par with IRED (Du et al., 2024), an energy-based diffusion model, and better than all other existing approaches (including the R-Transformer). *AKOrN* again strongly outperforms its non-Kuramoto counterparts, ItrSA and Transformer.

Test-time extension of the Kuramoto steps Just as we humans use more time to solve harder problems, *AKOrN*'s performance improves as we increase the number of Kuramoto steps. As shown in

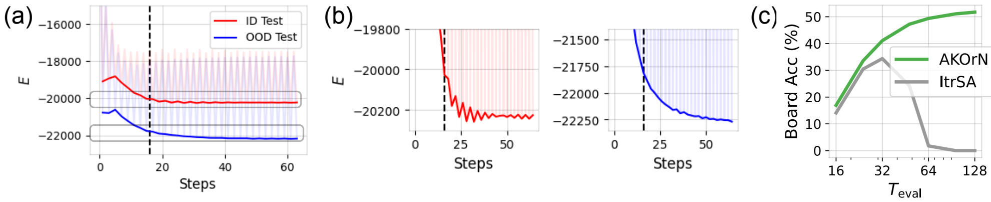


Figure 6: (a) Transition of the energy in Eq. (3) over # Kuramoto steps on the Sudoku datasets. The semi-transparent lines are actual energy values averaged across examples, and the solid ones connect the troughs. The dotted vertical line indicates # Kuramoto steps set during training. (b) A zoomed-in version of each plot. (c) The effect of test-time extension on # Kuramoto steps.

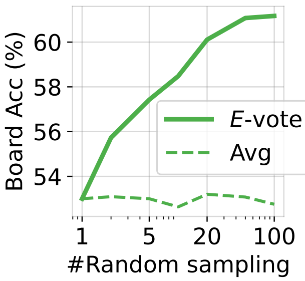


Figure 7: Improvement of board accuracy by the post-selection of predictions based on the E values described in Sec 6.2. T_{eval} is set to 128. 'E-vote' and 'Avg' stand for energy-based voting and majority voting, respectively.

Model	ID	OOD
SAT-Net (Wang et al., 2019)	98.3	3.2
Diffusion (Du et al., 2024)	66.1	10.3
IREM (Du et al., 2022)	93.5	24.6
RRN (Palm et al., 2018)	99.8	28.6
R-Transformer (Yang et al., 2023)	**100.0**	30.3
IRED (Du et al., 2024)	99.4	**62.1**
Transformer	98.6±0.3	5.2±0.2
ItrSA	95.7±8.5	34.4±5.4
AKOrN$^{\text{attn}}$	**100.0**±0.0	**61.1**±14.7

Table 3: Board accuracy on Sudoku Puzzles. We show the mean and std of the accuracy of models with 5 different random seeds for the weight initialization. The AKOrN results are obtained with $T_{\text{eval}} = 128$ and the energy-based voting with 100 samples of initial oscillators.

Fig. 6 (a,b), on the ID test set, the energy fluctuates but roughly converges to a minimum after around 32 steps. On the OOD test set, however, the energy continues to decrease further. Fig. 6 (c) shows that increasing the number of Kuramoto steps at test time improves accuracy significantly (17% to 52%), while increasing the step count of standard self-attention provides a limited improvement on the OOD set (14% to 34%) and leads to lower performance on the ID set (99.3% to 95.7%).

The energy value tells the correctness of the boards The energy value is a good indicator of the solution's correctness. In fact, we observe that predictions with low-energy oscillator states tend to be correct (see Fig. 26). We utilize this property to improve the performance. For each given board, we sample multiple predictions with different initial oscillators and select the lowest-energy prediction as the model's answer, which we call *Energy-based voting* (E-vote). We see in Fig. 7 that by increasing the number of sampled predictions, the model's board accuracy improves. Just averaging the predictions of different states (i.e., majority voting) does not give better answers.

6.3 ROBUSTNESS AND CALIBRATION

We test *AKOrN*'s robustness to adversarial attacks and its uncertainty quantification performance on CIFAR10 and CIFAR10 with common corruptions (CC, Hendrycks & Dietterich (2019)). We train two types of networks: a convolutional AKOrN (*AKOrN*$^{\text{conv}}$) and AKOrN with both convolution and self-attention (*AKOrN*$^{\text{mix}}$). The former has three convolutional Kuramoto layers. The latter replaces the last block with an attentive Kuramoto block. We use AutoAttack (Caron et al., 2021) to evaluate the model's adversarial robustness.

***AKOrN*s are resilient against gradient-based attacks** The model is heavily regularized and achieves both good adversarial robustness and robustness to natural corruptions (Tab. 4). This is remarkable, since conventional neural models need additional techniques such as adversarial training and/or adversarial purification to achieve good adversarial robustness. In contrast, *AKOrN* is robust by design, even when trained on only clean examples.

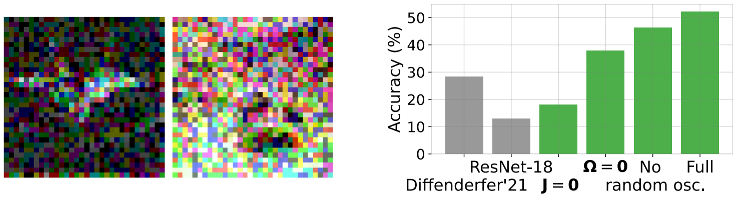


Figure 8: Robustness performance on random noise examples. Each bar plot shows classification accuracy on CIFAR10 with strong random noise ($\|\epsilon\|_\infty = 64/255$). The left two pictures are examples of images with that ϵ. Green bars show accuracy when we ablate each element of *AKOrN*.

Model	↑ Accuracy Clean	 Adv	 CC	↓ ECE CC
Bartoldson et al. (2024)	93.68	73.71	75.9	20.5
Diffenderfer et al. (2021)	96.56	0.00	92.8	4.8
ViT	91.44	0.00	81.0	9.6
ResNet-18	94.41	0.00	81.5	8.9
AKOrN$^{\text{conv}}$	88.91	*58.91	83.0	1.3
AKOrN$^{\text{mix}}$	91.23	*51.56	86.4	1.4

Table 4: Robustness to adversarial examples by AutoAttack (Adv) and common corruptions (CC) on CIFAR10. *The attack is done by AutoAttack with EoT (Athalye et al., 2018). $\|\epsilon\|_\infty$ is set to 8/255. Expected Calibration Error (ECE) measures the alignment between confidence of the prediction and accuracy. The top two methods are selected from the highest-ranked methods on https://robustbench.github.io/.

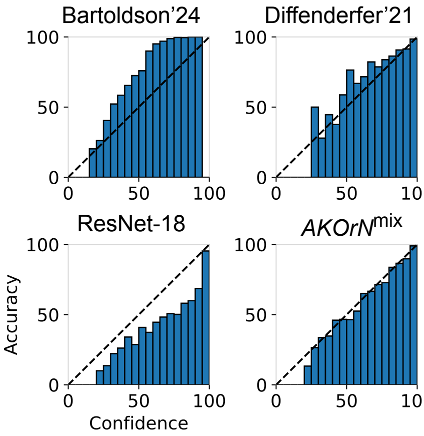


Figure 9: Confidence vs Accuracy plots on CIFAR10 with common corruptions.

K-Nets are well-calibrated and robust to strong random noise

We found that *AKOrN*s are robust to strong random noise (Fig. 8) and give good uncertainty estimation (on the bottom right in Fig. 9). Surprisingly, there is an almost perfect correlation between confidence and actual accuracy. This is similar to observations in generative models (Grathwohl et al., 2020; Jaini et al., 2024), where conditional generative models give well-calibrated outputs. Since *AKOrN*'s energy is not learned to model input distribution, we cannot tightly relate ours to such generative models. However, we speculate that *AKOrN*s' energy roughly approximates the likelihood of the input examples, and thus the oscillator state fluctuates according to the height of the energy, which would result in good calibration.

7 DISCUSSION & CONCLUSION

We propose *AKOrN*, which integrates the Kuramoto model into neural networks and scales to complex observations, such as natural images. *AKOrN*s learn strongly object-binding features, can reason, and are robust to adversarial and natural perturbations with well-calibrated predictions. We believe our work provides a foundation for exploring a fundamental shift in the current neural network paradigm.

In the current formulation of *AKOrN*, each oscillator is constrained onto the sphere and each single oscillator cannot represent the 'presence' of the features like the rotating features in Löwe et al. (2023). Because of that, *AKOrN* would not perform well on memory tasks, where the model needs to remember the presence of events. This norm constraint also does not align with real biological neurons that have firing and non-firing states. Relaxing the hard norm constraint of the oscillator would be an interesting future direction in terms of both biological plausibility and applicability to a much wider range of tasks such as long-term temporal processing.

REFERENCES

Shun-Ichi Amari and Michael A Arbib. Competition and cooperation in neural nets. *Systems neuroscience*, pp. 119–165, 1977.

Toshio Aoyagi. Network of neural oscillators for retrieving phase information. *Physical review letters*, 74(20): 4075, 1995.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proc. of the International Conf. on Machine learning (ICML)*, pp. 274–283, 2018.

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

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv.org*, 2014.

Brian R Bartoldson, James Diffenderfer, Konstantinos Parasyris, and Bhavya Kailkhura. Adversarial robustness limits via scaling-law and human-alignment studies. In *Proc. of the International Conf. on Machine learning (ICML)*, 2024.

Gabriel B Benigno, Roberto C Budzinski, Zachary W Davis, John H Reynolds, and Lyle Muller. Waves traveling over a map of visual space can ignite short-term predictions of sensory input. *Nature Communications*, 14(1):3409, 2023.

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Ondrej Biza, Sjoerd Van Steenkiste, Mehdi SM Sajjadi, Gamaleldin F Elsayed, Aravindh Mahendran, and Thomas Kipf. Invariant slot attention: Object discovery with slot-centric reference frames. In *Proc. of the International Conf. on Machine learning (ICML)*, 2023.

Wilfried Bounsi, Borja Ibarz, Andrew Dudzik, Jessica B Hamrick, Larisa Markeeva, Alex Vitvitskyi, Razvan Pascanu, and Petar Veličković. Transformers meet neural algorithmic reasoners. *arXiv.org*, 2024.

Michael Breakspear, Stewart Heitmann, and Andreas Daffertshofer. Generative models of cortical oscillations: neurobiological implications of the kuramoto model. *Frontiers in human neuroscience*, 4:190, 2010.

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

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, pp. 9650–9660, 2021.

Sarthak Chandra, Michelle Girvan, and Edward Ott. Continuous versus discontinuous transitions in the d-dimensional generalized kuramoto model: Odd d is different. *Physical Review X*, 9(1):011002, 2019.

Michael Chang, Tom Griffiths, and Sergey Levine. Object representations as fixed points: Training iterative refinement algorithms with implicit differentiation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 32694–32708, 2022.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proc. of the International Conf. on Machine learning (ICML)*, pp. 1597–1607, 2020.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, pp. 9640–9649, 2021.

Zachary W Davis, Lyle Muller, Julio Martinez-Trujillo, Terrence Sejnowski, and John H Reynolds. Spontaneous travelling cortical waves gate perception in behaving primates. *Nature*, 587(7834):432–436, 2020.

Bhishma Dedhia and Niraj K Jha. Neural slot interpreters: Grounding object semantics in emergent slot representations. *arXiv.org*, 2024.

James Diffenderfer, Brian Bartoldson, Shreya Chaganti, Jize Zhang, and Bhavya Kailkhura. A winning hand: Compressing deep networks can improve out-of-distribution robustness. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 664–676, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2021.

Yilun Du, Shuang Li, Joshua Tenenbaum, and Igor Mordatch. Learning iterative reasoning through energy minimization. In *Proc. of the International Conf. on Machine learning (ICML)*, pp. 5570–5582, 2022.

Yilun Du, Jiayuan Mao, and Joshua B Tenenbaum. Learning iterative reasoning through energy diffusion. In *Proc. of the International Conf. on Machine learning (ICML)*, 2024.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. Faith and fate: Limits of transformers on compositionality. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Felix Effenberger, Pedro Carvalho, Igor Dubinin, and Wolf Singer. The functional role of oscillatory dynamics in neocortical circuits: a computational perspective. *bioRxiv*, pp. 2022–11, 2022.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision (IJCV)*, 88:303–338, 2010.

Juergen Fell and Nikolai Axmacher. The role of phase synchronization in memory processes. *Nature reviews neuroscience*, 12(2):105–118, 2011.

Michel Fruchart, Ryo Hanai, Peter B Littlewood, and Vincenzo Vitelli. Non-reciprocal phase transitions. *Nature*, 592(7854):363–369, 2021.

Borjan Geshkovski, Cyril Letrouit, Yury Polyanskiy, and Philippe Rigollet. The emergence of clusters in self-attention dynamics. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2014.

Anand Gopalakrishnan, Aleksandar Stanić, Jürgen Schmidhuber, and Michael Curtis Mozer. Recurrent complex-weighted autoencoders for unsupervised object discovery. *arXiv.org*, 2024.

Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv.org*, 2020.

Sven Gowal, Sylvestre-Alvise Rebuffi, Olivia Wiles, Florian Stimberg, Dan Andrei Calian, and Timothy A Mann. Improving robustness using generated data. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 4218–4233, 2021.

Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2020.

Charles M Gray, Peter König, Andreas K Engel, and Wolf Singer. Oscillatory responses in cat visual cortex exhibit inter-columnar synchronization which reflects global stimulus properties. *Nature*, 338(6213):334–337, 1989.

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 *Proc. of the International Conf. on Machine learning (ICML)*, pp. 2424–2433, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pp. 630–645, 2016.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 16000–16009, 2022.

Werner Heisenberg. *Zur theorie des ferromagnetismus*. Springer, 1985.

Stewart Heitmann, Pulin Gong, and Michael Breakspear. A computational role for bistability and traveling waves in motor cortex. *Frontiers in computational neuroscience*, 6:67, 2012.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2019.

Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2020.

Sara Hooker. The hardware lottery. *Communications of the ACM*, 64(12):58–65, 2021.

Benjamin Hoover, Yuchen Liang, Bao Pham, Rameswar Panda, Hendrik Strobelt, Duen Horng Chau, Mohammed Zaki, and Dmitry Krotov. Energy transformer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106, 1962.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. of the International Conf. on Machine learning (ICML)*, 2015.

Priyank Jaini, Kevin Clark, and Robert Geirhos. Intriguing properties of generative classifiers. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2024.

Baoxiong Jia, Yu Liu, and Siyuan Huang. Improving object-centric learning with query optimization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2023.

Jindong Jiang, Fei Deng, Gautam Singh, and Sungjin Ahn. Object-centric slot diffusion. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Whie Jung, Jaehoon Yoo, Sungjin Ahn, and Seunghoon Hong. Learning to compose: Improving object centric learning by injecting compositionality. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2024.

Rishabh Kabra, Chris Burgess, Loic Matthey, Raphael Lopez Kaufman, Klaus Greff, Malcolm Reynolds, and Alexander Lerchner. Multi-object datasets. https://github.com/deepmind/multi-object-datasets/, 2019.

Ioannis Kakogeorgiou, Spyros Gidaris, Konstantinos Karantzalos, and Nikos Komodakis. Spot: Self-training with patch-order permutation for object-centric learning with autoregressive transformers. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 22776–22786, 2024.

Laurynas Karazija, Iro Laina, and Christian Rupprecht. Clevrtex: A texture-rich benchmark for unsupervised multi-object segmentation. *arXiv.org*, 2021.

T Anderson Keller, Lyle Muller, Terrence J Sejnowski, and Max Welling. A spacetime perspective on dynamical computation in neural information processing systems. *arXiv.org*, 2024.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2015.

Klim Kireev, Maksym Andriushchenko, and Nicolas Flammarion. On the effectiveness of adversarial training against common corruptions. In *Uncertainty in Artificial Intelligence*, pp. 1012–1021. PMLR, 2022.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

Yoshiki Kuramoto. *Chemical turbulence*. Springer, 1984.

Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *International journal of computer vision*, 128(7):1956–1981, 2020.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Nicolas Le Roux, Nicolas Heess, Jamie Shotton, and John Winn. Learning a generative model of images by factoring appearance and shape. *Neural Computation*, 23(3):593–650, 2011.

Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *arXiv.org*, 2017.

Alexander C Li, Mihir Prabhudesai, Shivam Duggal, Ellis Brown, and Deepak Pathak. Your diffusion model is secretly a zero-shot classifier. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, pp. 2206–2217, 2023.

Luisa HB Liboni, Roberto C Budzinski, Alexandra N Busch, Sindy Löwe, Thomas A Keller, Max Welling, and Lyle E Muller. Image segmentation with traveling waves in an exactly solvable recurrent neural network. *arXiv.org*, 2023.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pp. 740–755. Springer, 2014.

Max Lipton, Renato Mirollo, and Steven H Strogatz. The kuramoto model on a sphere: Explaining its low-dimensional dynamics with group theory and hyperbolic geometry. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(9), 2021.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 11525–11538, 2020.

Sindy Löwe, Phillip Lippe, Maja Rudolph, and Max Welling. Complex-valued autoencoders for object discovery. *arXiv preprint arXiv:2204.02075*, 2022.

Sindy Löwe, Phillip Lippe, Francesco Locatello, and Max Welling. Rotating features for object discovery. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Sindy Löwe, Francesco Locatello, and Max Welling. Binding Dynamics in Rotating Features. *ICLR 2024 Workshop: Bridging the Gap Between Practice and Theory in Deep Learning*, 2024.

Evgueniy V Lubenov and Athanassios G Siapas. Hippocampal theta oscillations are travelling waves. *Nature*, 459(7246):534–539, 2009.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv.org*, 2017.

Daniel C Mattis. *The theory of magnetism I: Statics and Dynamics*, volume 17. Springer Science & Business Media, 2012.

Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

Takeru Miyato, Bernhard Jaeger, Max Welling, and Andreas Geiger. Gta: A geometry-aware attention mechanism for multi-view transformers. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2024.

Tom Monnier, Elliot Vincent, Jean Ponce, and Mathieu Aubry. Unsupervised layered image decomposition into object prototypes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 8640–8650, 2021.

Vernon B Mountcastle. The columnar organization of the neocortex. *Brain: a journal of neurology*, 120(4): 701–722, 1997.

Lyle Muller, Giovanni Piantoni, Dominik Koller, Sydney S Cash, Eric Halgren, and Terrence J Sejnowski. Rotating waves during human sleep spindles organize global patterns of activity that repeat precisely through the night. *Elife*, 5:e17267, 2016.

Lyle Muller, Frédéric Chavane, John Reynolds, and Terrence J Sejnowski. Cortical travelling waves: mechanisms and computational principles. *Nature Reviews Neuroscience*, 19(5):255–268, 2018.

Andrew Ng and Michael Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2001.

Reza Olfati-Saber. Swarms on sphere: A programmable swarm with synchronous behaviors like oscillator networks. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pp. 5060–5066. IEEE, 2006.

Rasmus Palm, Ulrich Paquet, and Ole Winther. Recurrent relational networks. *Advances in neural information processing systems*, 31, 2018.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, et al. Hopfield networks is all you need. *arXiv.org*, 2020.

David P Reichert and Thomas Serre. Neuronal synchrony in complex-valued deep networks. *arXiv preprint arXiv:1312.6115*, 2013.

Matthew Ricci, Minju Jung, Yuwei Zhang, Mathieu Chalvidal, Aneri Soni, and Thomas Serre. Kuranet: systems of coupled oscillators that learn to synchronize. *arXiv.org*, 2021.

James A Roberts, Leonardo L Gollo, Romesh G Abeysuriya, Gloria Roberts, Philip B Mitchell, Mark W Woolrich, and Michael Breakspear. Metastable brain waves. *Nature communications*, 10(1):1056, 2019.

Doug Rubino, Kay A Robbins, and Nicholas G Hatsopoulos. Propagating waves mediate information transfer in the motor cortex. *Nature neuroscience*, 9(12):1549–1557, 2006.

Bruno Sauvalle and Arnaud de La Fortelle. Unsupervised multi-object segmentation using attention and soft-argmax. In *Proc. of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 3267–3276, 2023.

Maximilian Seitzer, Max Horn, Andrii Zadaianchuk, Dominik Zietlow, Tianjun Xiao, Carl-Johann Simon-Gabriel, Tong He, Zheng Zhang, Bernhard Schölkopf, Thomas Brox, et al. Bridging the gap to real-world object-centric learning. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2023.

Charles Scott Sherrington. The integrative action of the nervous system. *Yale University Press*, 1906.

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

Gautam Singh, Yi-Fu Wu, and Sungjin Ahn. Simple unsupervised object-centric learning for complex and naturalistic videos. pp. 18181–18196, 2022.

David C Somers, Sacha B Nelson, and Mriganka Sur. An emergent model of orientation selectivity in cat visual cortical simple cells. *Journal of neuroscience*, 15(8):5448–5465, 1995.

Aleksandar Stanić, Anand Gopalakrishnan, Kazuki Irie, and Jürgen Schmidhuber. Contrastive training of complex-valued autoencoders for object discovery. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv.org*, 2021.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2014.

Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1633–1645, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 5998–6008, 2017.

Po-Wei Wang, Priya Donti, Bryan Wilder, and Zico Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *Proc. of the International Conf. on Machine learning (ICML)*, pp. 6545–6554, 2019.

Tianshi Wang and Jaijeet Roychowdhury. Oscillator-based ising machine. *arXiv.org*, 2017.

Yuxin Wu and Kaiming He. Group normalization. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pp. 3–19, 2018.

Ziyi Wu, Jingyu Hu, Wuyue Lu, Igor Gilitschenski, and Animesh Garg. Slotdiffusion: Object-centric generative modeling with diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 50932–50958, 2023.

Zhun Yang, Adam Ishay, and Joohyung Lee. Learning to solve constraint satisfaction problems with recurrent transformer. *arXiv.org*, 2023.

Honghui Zhang, Andrew J Watrous, Ansh Patel, and Joshua Jacobs. Theta and alpha oscillations are traveling waves in the human neocortex. *Neuron*, 98(6):1269–1281, 2018.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *Proc. of the International Conf. on Machine learning (ICML)*, pp. 7472–7482, 2019.

Hao Zheng, Hui Lin, and Rong Zhao. Gust: combinatorial generalization by unsupervised grouping with neuronal coherence. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2023.

Jiandong Zhu. Synchronization of kuramoto model in a high-dimensional linear space. *Physics Letters A*, 377 (41):2939–2943, 2013.

Appendix

Table of Contents

A Model analysis **19**
- A.1 Projection operator . . . 19
- A.2 Replacing the Kuramoto model with the conventional residual update . . . 20
- A.3 Number of rotating dimensions . . . 20
- A.4 The bias term $\mathbf{C}$ and norm-taking term $\mathbf{m}$ . . . 21
- A.5 Training & inference time . . . 21

B Relation to physics models **21**

C Related works on the NN robustness **22**

D Experimental settings **22**
- D.1 Unsupervised object discovery . . . 22
- D.2 Sudoku solving . . . 26
- D.3 Robustness and calibration on CIFAR10 . . . 27

E Additional experimental results **29**
- E.1 Positional encoding for the attentive connectivity . . . 29
- E.2 Unsupervised object discovery . . . 30
- E.3 Sudoku solving . . . 34
- E.4 Symmetric constraint . . . 35
- E.5 Robustness and calibration on CIFAR10 . . . 36

F Additional cluster visualizations **37**

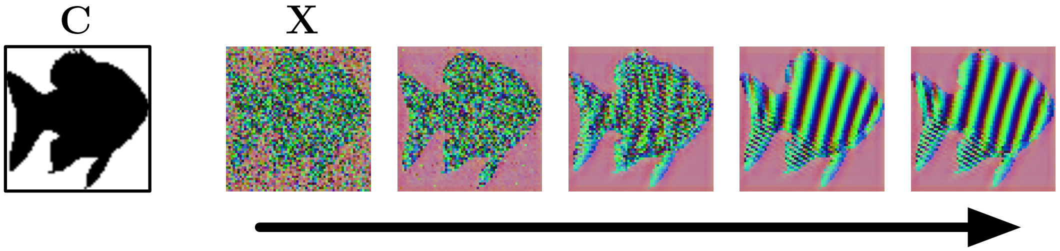
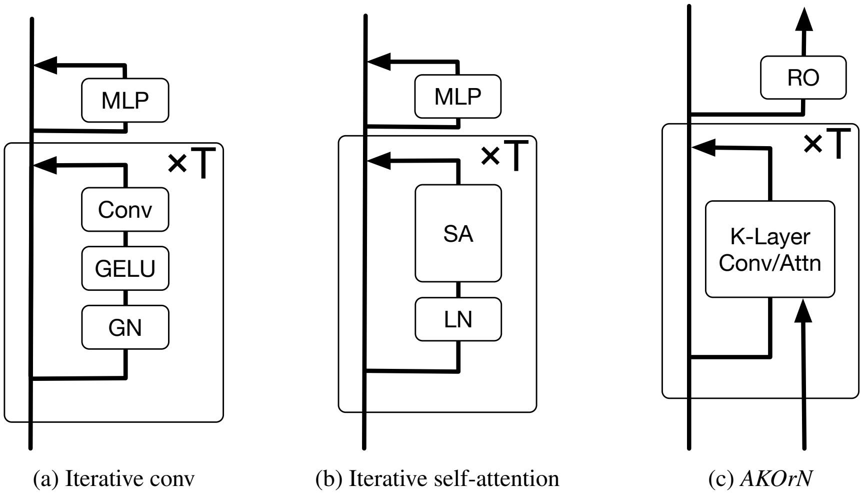


Figure 10: The transition of the 64×64 oscillator neurons ($N = 4$). (Left) Visualiaztion of $\mathbf{C}$. $\mathbf{c}_i$ on the white region is set to $\mathbf{1}$ and the black region is set to $\mathbf{0}$. (Right) Oscillators' time evolution. Similar colors indicate oscillators directing similar directions. The connectivity $\mathbf{J}$ is a 9×9 convolution kernel with random filters. The oscillators on the white region of $\mathbf{C}$ are aligned with the conditional stimuli and almost stay constant across time. The oscillators on the black region are largely influenced by the neighboring oscillators and exhibit wavy patterns.



(a) Iterative conv (b) Iterative self-attention (c) *AKOrN*

Figure 11: Block diagrams of (a) ItrConv (b) ItrSA, and (c) *AKOrN*. GN and LN stand for Group Normalization (Wu & He, 2018) and Layer normalization (Ba et al., 2016), respectively. The MLP in (a) or (b) is composed of a stack of GN or LN followed by Linear, GELU, and Linear layers. The hidden dim of MLP is set to 2× (channel size). The number of heads in SA and the K-Layer with attentive connectivity is set to 8 throughout our experiments.

A MODEL ANALYSIS

In this section, we provide an extensive comparison of the architectural designs. Specifically, we show:

- A visual description of the projection operator and its effect on the performance (Sec. A.1)
- The Kuramoto model vs conventional residual update (Sec. A.2)
- The effect of the number of rotating dimensions N (Sec. A.3)
- The efficacy of $\mathbf{C}$ and $\mathbf{m}$ in *AKOrN* (Sec. A.4).

Additionally, we show run-time comparisons between *AKOrN*s and their non-Kuramoto counterparts on different datasets in Sec. A.5.

A.1 PROJECTION OPERATOR

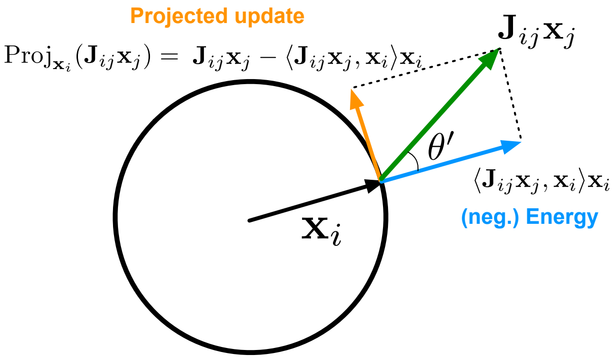

Projected update
$\mathrm{Proj}_{\mathbf{x}_i}(\mathbf{J}_{ij}\mathbf{x}_j) = \mathbf{J}_{ij}\mathbf{x}_j - \langle \mathbf{J}_{ij}\mathbf{x}_j, \mathbf{x}_i\rangle \mathbf{x}_i$
$\mathbf{J}_{ij}\mathbf{x}_j$
θ'
$\langle \mathbf{J}_{ij}\mathbf{x}_j, \mathbf{x}_i\rangle \mathbf{x}_i$
(neg.) Energy
$\mathbf{x}_i$

Figure 12: Visual description of $\mathrm{Proj}_{\mathbf{x}_i}(\mathbf{J}_{ij}\mathbf{x}_j)$. θ' is the angle difference between $\mathbf{x}_i$ and $\mathbf{J}_{ij}\mathbf{x}_j$. Given the length of $\mathbf{J}_{ij}\mathbf{x}_j$, the length of the projected update in Eq. (2) and the negative energy in Eq. (3) are inversely proportional.

Fig. 12 illustrates a visual representation of the projection operator $\mathrm{Proj}_{\mathbf{x}_i}$ defined in Eq. (2). Note that this operator plays a key role in Riemannian optimization on the sphere, ensuring that the updated direction lies within the tangent space at the point $\mathbf{x}_i$ on the sphere.

Relation between the vectorized Kuramoto model and the original one The vectorized Kuramoto model includes the original one in a special case. Suppose the case of $N = 2$, $\mathbf{c}_i = \mathbf{0}$, and having a scalar connection for $\mathbf{J}_{ij}$: $\mathbf{J}_{ij} = J_{ij}\mathbf{I}$ where $J_{ij} \in \mathbb{R}$ and $\mathbf{I}$ is the 2×2 identity matrix. Then we have $\theta' = \theta_j - \theta_i$ where $\theta_i, \theta_j = \arg(\mathbf{x}_i), \arg(\mathbf{x}_j)$. From the definition of trigonometric functions, we get

$$\langle \mathbf{J}_{ij}\mathbf{x}_j, \mathbf{x}_i\rangle \mathbf{x}_i = J_{ij}\cos(\theta_j - \theta_i)\mathbf{x}_i \tag{10}$$

$$\mathrm{Proj}_{\mathbf{x}_i}(\mathbf{x}_j) = \mathbf{J}_{ij}\mathbf{x}_j - \langle \mathbf{J}_{ij}\mathbf{x}_j, \mathbf{x}_i\rangle \mathbf{x}_i = J_{ij}\sin(\theta_j - \theta_i)\mathbf{x}_i^{\perp}, \tag{11}$$

where $\mathbf{x}_i^{\perp}$ is the unit vector perpendicular to $\mathbf{x}_i$ and its direction is increasing θ_i. Thus the Eq. (2) is an extension of Eq. (1). This proof is just a rephrased version of Chandra et al. (2019) and Proposition 1 in Olfati-Saber (2006). Please refer to them for details.

Note that with or without Proj only changes the length of the update direction of each neuron. The updated $\mathbf{x}_i$ stays on the sphere since we normalize each updated neuron to be the unit vector in Eq. (5). We test AKOrN without Proj operators and summarize the results in Tab. 5. We see almost identical and a bit degraded performance on the CLEVR-TEx object discovery and Sudoku solving, respectively. Interestingly, without projection, the adversarial robustness and uncertainty quantification get worse than the original *AKOrN*.

$\mathrm{Proj}_{\mathbf{x}}$	FG-ARI	MBO
✗	80.9	**57.4**
✓	**81.5**	54.1

(a) CLEVR-Tex

$\mathrm{Proj}_{\mathbf{x}}$	ID	OOD
✗	99.9±0.0	45.0±1.9
✓	**100.0**±0.0	**51.7**±3.3

(b) Sudoku

	↑ Accuracy			↓ ECE
$\mathrm{Proj}_{\mathbf{x}}$	Clean	Adv	CC	CC
✗	**89.9**	0.1	**82.4**	4.5
✓	84.6	**64.9**	78.3	**1.8**

(c) CIFAR10

Table 5: Ablation of $\mathrm{Proj}_{\mathbf{x}}$.

A.2 REPLACING THE KURAMOTO MODEL WITH THE CONVENTIONAL RESIDUAL UPDATE

Here, we conduct an ablation study of the Kuramoto updates. Specifically, we train a proposed *AKOrN* architecture on CLEVRTex and Sudoku, but without projection and normalization (Proj and Π in Eqs (4) and (5)). The update results in the conventional residual update. Tab. 6 shows that the ablated model degrades both the object discovery performance and Sudoku solving significantly, which clearly shows the large contribution of the Kuramoto update to the performances.

Kuramoto	FG-ARI	MBO
✗	65.0	51.3
✓	**81.5**	**54.1**

(a) CLEVR-Tex

Kuramoto	ID	OOD
✗	59.8±54.6	17.1±16.6
✓	**100.0**±0.0	**51.7**±3.3

(b) Sudoku

Table 6: With or without the use of Kuramoto updates.

A.3 NUMBER OF ROTATING DIMENSIONS

N	FG-ARI	MBO
2	47.6	28.8
4	**81.5**	**54.1**

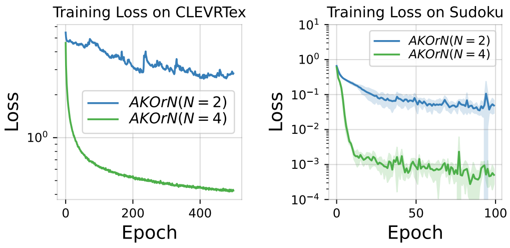
(a) CLEVR-Tex

N	ID	OOD
2	0.3±0.4	0.0±0.0
4	**100.0**±0.0	**51.7**±3.3

(b) Sudoku

Table 7: The effect of the number of rotating dimensions. The inferior performance with $N = 2$ comes from the model's underfitting (See the next figures)

.



(c) CLEVR-Tex (d) Sudoku

Figure 14: Loss curve comparison between models with different N.

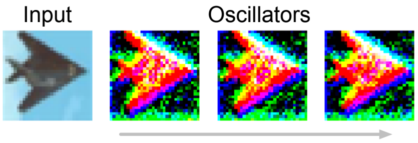


Figure 15: Noisy oscillators when $N = 2$. The three panels show the states of the oscillators at consecutive time steps. The animation file cifar10_block1.gif in the Supplementary Material provides more clear visualization of the fluctuations.

	↑ Accuracy			↓ ECE
Model	Clean	Adv	CC	CC
ResNet ($\sigma = 0.2$)	85.2	22.3	75.1	2.3
ResNet ($\sigma = 0.225$)	83.9	25.5	73.6	2.6
AKOrN ($N = 2$)	84.6	**64.9**	**78.3**	**1.8**

Table 8: Robustness comparison with ResNets trained to resist Gaussian noises. σ indicates the standard deviation of the noise added during training.

A.4 THE BIAS TERM $\mathbf{C}$ AND NORM-TAKING TERM $\mathbf{m}$

AKOrN employs a two-stream architecture to process observations, which helps stabilize training. Additionally, the norm-taking part $\mathbf{m}$ in Eq. (6) plays a key role in improving the model's fitness to the data. Fig. 16 presents a performance comparison with a model stacking only K-layers (Staking K-Layers) and *AKOrN* without $\mathbf{m}$. Here, 'Stacking K-Layers' removes the term $\mathbf{C}$ in Eq. (4) and instead processes an observation into $\mathbf{X}^{(0)}$ by using a single 3×3 convolution applied to the RGB input. We see the use of $\mathbf{C}$ and $\mathbf{m}$ significantly contributes to the loss minimization. The final test accuracies of Stacking K-Layers, *AKOrN* wo/ $\mathbf{m}$, and the original *AKOrN* are 62.9, 77.8, and 84.6, respectively.

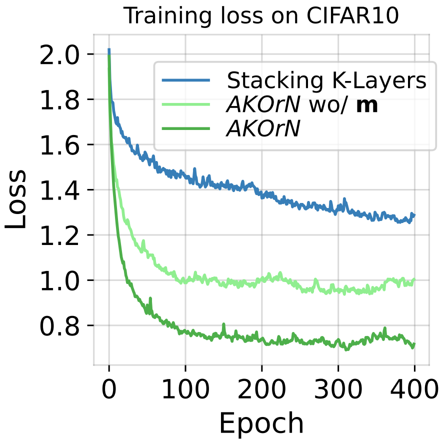
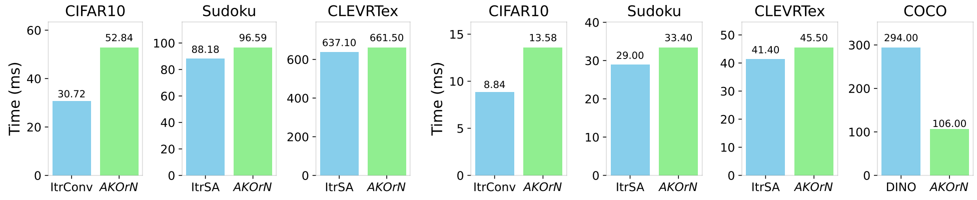


Figure 16: Effect of the use of $\mathbf{C}$ and $\mathbf{m}$ in Eq. (6)

A.5 TRAINING & INFERENCE TIME



(a) Training time

(b) Inference time

Figure 17: Training and inference time in different tasks. Training time is the time taken to complete a single gradient step (excluding data loading). Inference time is the time taken for a single forward pass with a mini-batch size of 100.

B RELATION TO PHYSICS MODELS

Similar to how the Ising model is the basis for recurrent neural models, such as the Hopfield model (Hopfield, 1982), the Kuramoto model with symmetric lateral interactions can also be studied by viewing it as a model from statistical physics called the Heisenberg model (Mattis, 2012). In fact, we use more general version of the Kuramoto model which involves a symmetry-breaking term (akin to a magnetic field interaction) and asymmetric connections between the neurons. This not only is biologically plausible (synapses are not symmetric), it also leads to much better results in our experiments.

Non-equilibrium soft matter physics has studied models with nonreciprocal interactions, for instance in the field of "active matter". They have developed accurate coarse-grained hydrodynamics models to approximate the microscopic dynamics and observed very interesting behavior, such as symmetry-breaking phase transitions and resultant traveling waves representing so-called Goldstone modes (Fruchart et al., 2021). We hope that this opens the door to a deeper understanding of these models when employed as neural networks.

C RELATED WORKS ON THE NN ROBUSTNESS

Experimental proof of the conventional NNs' limited OOD generalization is represented by the vulnerability to adversarial examples (Szegedy et al., 2014; Goodfellow et al., 2014). The most effective way to resist such examples is training the model on adversarial examples generated by the model itself, which is called adversarial training (Goodfellow et al., 2014; Madry et al., 2017; Miyato et al., 2018; Zhang et al., 2019). Many other defenses have been proposed, but most of them were found to be not a fundamental solution (Tramer et al., 2020).

One framework that can produce more human-algined predictions is a generative classifier (Ng & Jordan, 2001; Bishop & Nasrabadi, 2006), where we train a model with both generative and discriminative objectives or turn a label conditional generative model into a discriminative model based on Bayes theorem. Interestingly, different generative classifiers trained with different methods share similar robust and calibration properties (Lee et al., 2017; Grathwohl et al., 2020; Li et al., 2023; Jaini et al., 2024). Generative classifiers are robust but involve costly generative training such as denoising diffusion (Li et al., 2023; Jaini et al., 2024), MCMC (Grathwohl et al., 2020) to generate negative samples, or unstable min-max optimization as GANs training (Lee et al., 2017). *AKOrN* shares similar robustness properties but without any generative objectives.

D EXPERIMENTAL SETTINGS

We observe that both the readout module and conditional stimuli $\mathbf{C}$ are essential for stable training, especially when $N = 2$. We also see that *AKOrN* with $N = 2$ exhibits a strong regularity, which acts positively on robustness performance while having negative effects on unsupervised object discovery and the Sudoku-solving experiments. We show results of *AKOrN* with $N = 4$ in those experiments. We do not observe improvement by increasing N above 4. Further experimental and mathematical analysis is needed to understand why this occurs, which could provide insights into how we can leverage both advantages.

Tabs 9-12 show experimental settings on each dataset (e.g. hyperparameters on models and optimization, the number of training and test examples, dataset statistics, etc...). For *AKOrN*, the channel size is set to (the channel size shown in the table)$/N$, so that the memory consumption and FLOPs are effectively the same between *AKOrN*s and their non-Kuramoto counterpart baselines. All models are trained with Adam (Kingma & Ba, 2015) without weight decay.

D.1 UNSUPERVISED OBJECT DISCOVERY

We test on 4 synthetic benchmark datasets (Tetrominoes, dSprites, CLEVR, CLEVRTex), one synthetic dataset created by us (Shapes), and 2 real image benchmark datasets (PascalVOC, COCO2017). The Shapes dataset consists of images with 2–4 objects that are randomly sampled from four basic shapes (triangle, square, circle, and diamond). Note that each image can have multiple objects of the same shape together.

The kernel size of convolution layers in *AKOrN*$^{\text{conv}}$ and ItrConv is set to 5, 7, and 9 on Tetrominoes, dSprites, and CLEVR, respectively. In addition to ItrConv and ItrSA, we also train a ViT model (Dosovitskiy et al., 2021) as another baseline.

All networks process images similarly to ViT (Dosovitskiy et al., 2021). First, we patch each image into $H/P \times W/P$ patches where H, W are the height and width of the image and P is the patch size. We then apply the stack of blocks. The output of the final layer is further processed by global max-pooling followed by a single hidden layer MLP, whose output is used to compute the SimCLR loss. We used a conventional set of augmentations for SSL training: random resizing, cropping, and color jittering. We also apply horizontal flipping for the ImageNet pretraining. All models including baseline models have roughly the same number of parameters and are trained with shared

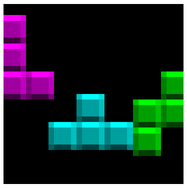
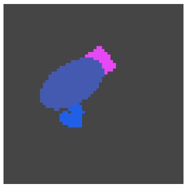
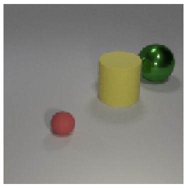
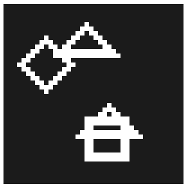
	Tetrominoes	dSprites	CLEVR	Shapes
				
Training examples	60,000	60,000	50,000	40,000
Test examples	320	320	320	1,000
Image size	32	64	128	40
Max. #objects	3	6	6	4
Patch size	4	4	8	2
Patch resolution	8	16	16	20
Channel size	128	128	256	256
#internal steps (T)	8	8	8	8/4
#Epochs	50	50	300	100
Batchsize	256			
Learning rate	0.001			
Augmentations	Random resize and crop + color jittering			
#clusters set for eval	4	7	11	5

Table 9: Experimental settings on Tetrominoes, dSprites, CLEVR, and Shapes. On Shapes, the internal steps are set to 8 for a single-block model and 4 for multiple-block models.

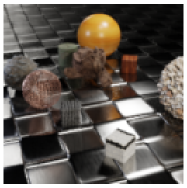
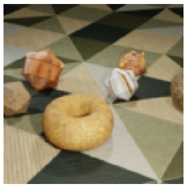
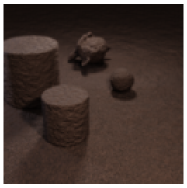
	CLEVRTex	OOD	CAMO
			
Training examples	40,000	-	-
Test examples	5,000	10,000	2,000
Image size	128		
Maximum number of objects	10		
Patch size	8		
Patch resolution	16		
Channel size	256		
# internal steps (T)	8		
# epochs	500	-	-
Batchsize	256	-	-
Learning rate	0.0005	-	-
Augmentations	Random resize and crop + color jittering		
#clusters set for eval	11		

Table 10: Experimental settings on CLEVRTex and its variants (OOD, CAMO). We also train a large *AKOrN* model that is trained with the doubled channel size, and epochs. We denote that model by Large *AKOrN*.

hyperparameters such as learning rates and training epochs. See Tabs 9-11 for those hyperparameter details.

	ImageNet	PascalVOC	COCO2017
Training examples	1,281,167	-	-
Test examples	-	1,449	5,000
Image size	256	256	256
Patch size		16	
Patch resolution		16	
Channel size		768	
# Blocks		3	
# internal steps (T)		4	
# epochs	400	-	-
Batchsize	512	-	-
Learning rate	0.0005	-	-
#clusters set for eval	-	4	7

Table 11: Experimental settings on ImageNet pratraining and on the PascalVOC and COCO2017 evaluation. For SimCLR training augmentations, we use random resize and crop, color jittering, and horizontal flipping.

<table>
<tr><th></th><th>Sudoku(ID) (Wang et al., 2019)</th><th>Sudoku(OOD) (Palm et al., 2018)</th></tr>
<tr><td></td><td><table>
<tr><td>9</td><td></td><td>1</td><td>5</td><td></td><td>3</td><td></td><td>6</td><td></td></tr>
<tr><td>3</td><td>6</td><td></td><td>2</td><td></td><td></td><td></td><td>1</td><td>8</td></tr>
<tr><td>2</td><td></td><td>7</td><td></td><td>4</td><td>6</td><td></td><td></td><td>9</td></tr>
<tr><td></td><td></td><td>4</td><td></td><td>7</td><td>2</td><td></td><td>5</td><td></td></tr>
<tr><td>1</td><td></td><td>9</td><td>3</td><td></td><td></td><td>8</td><td>4</td><td></td></tr>
<tr><td></td><td>7</td><td></td><td>8</td><td></td><td></td><td>9</td><td></td><td></td></tr>
<tr><td>6</td><td>5</td><td></td><td></td><td></td><td></td><td></td><td></td><td>4</td></tr>
<tr><td>4</td><td></td><td>8</td><td></td><td></td><td>9</td><td>3</td><td>7</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td>5</td><td>1</td><td>2</td><td></td><td></td></tr>
</table></td><td><table>
<tr><td></td><td></td><td></td><td></td><td>5</td><td></td><td></td><td>1</td><td></td></tr>
<tr><td></td><td>6</td><td></td><td>9</td><td></td><td></td><td>4</td><td></td><td>3</td></tr>
<tr><td></td><td></td><td>7</td><td></td><td></td><td>2</td><td></td><td></td><td></td></tr>
<tr><td>5</td><td>8</td><td></td><td></td><td></td><td></td><td></td><td>9</td><td></td></tr>
<tr><td></td><td></td><td></td><td>7</td><td></td><td></td><td>6</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td>3</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td></td><td></td><td></td><td>9</td><td></td><td>2</td><td></td><td></td></tr>
<tr><td>9</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>7</td></tr>
</table></td></tr>
<tr><td>Training examples</td><td>9,000</td><td>-</td></tr>
<tr><td>Test examples</td><td>1,000</td><td>18,000</td></tr>
<tr><td>Channel size</td><td colspan="2">512</td></tr>
<tr><td># epochs</td><td>100</td><td>-</td></tr>
<tr><td>Batchsize</td><td>100</td><td>-</td></tr>
<tr><td>Learning rate</td><td>0.0005</td><td>-</td></tr>
</table>

Table 12: Sudoku puzzle datasets.

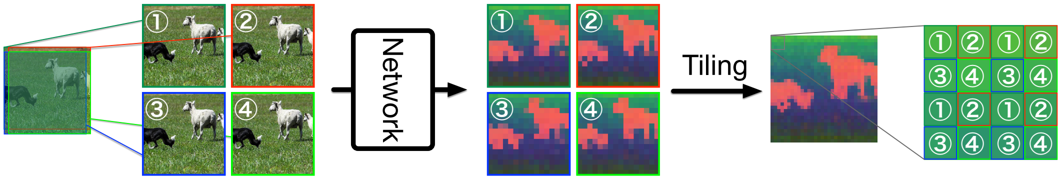


Figure 18: 2× *up-tiling*. First, we create horizontally or/and vertically shifted images with stride equal to (patchsize/2) and compute the model's output on each shifted image. We then interleave each token feature to make a 2× upsampled feature map.

In *AKOrN*, $\mathbf{C}^{(0)}$ is computed by the patched features of images, while each $\mathbf{x}_i$ is initialized by random oscillators sampled from the uniform distribution on the sphere. We use the identity function for g in each readout module. In multi-block models, we apply Group Normalization (Wu & He, 2018) to $\mathbf{C}$ except for the last block's output $\mathbf{C}^{(L)}$.

For the Tetrominoes, dSprites, and CLEVR datasets, we train single-block models with $T = 8$. We observe that stacking multiple blocks does not yield improvements on those three datasets. On CLEVRTex, we train single- and two-block models with attentive connectivity and $T = 8$, while on ImageNet, we train a three-block *AKOrN* model with attentive connectivity and $T = 4$.

D.1.1 METRICS

We use FG-ARI and MBO to evaluate cluster assignments, both of which are well-used metrics in object discovery tasks. We summarize

- FG-ARI: The Adjusted Rand Index (ARI) computes how well the clusters align with object masks compared to random cluster assignments. The foreground ARI (FG-ARI) only considers foreground objects and is a well-used metric in object discovery tasks. The maximum value of 100 indicates perfect alignment between the obtained clusters and the object masks. If the cluster assignment is completely random or all features are assigned to the same cluster, the value is 0.
- MBO: The Mean Best Overlap (MBO) first assigns each cluster to the highest overlapping ground truth mask and then computes the average intersection-over-union (IoU) of all pairs. The value takes 100 at maximum. Following the literature, we exclude the background mask from the MBO evaluation. Since MBO computes IoU, tightly aligned object masks give a higher value than FG-ARI (FG-ARI does not penalize the mask extending into the background region).

D.1.2 UPSAMPLE FEATURES BY UP-TILING

When we compute the cluster assignment, we upsample the output features by *up-tiling* where we let the model see a set of pictures that are slightly shifted both on the horizontal or/and vertical axes and make a higher resolution feature map by interleaving those features. This up-tiling enables us to get finer cluster assignments and substantially improves the object discovery performance of our *AKOrN*. We show a pictorial explanation in Fig. 18 and PyTorch code in Code 1. In Fig. 19, we compare up-tiled features with the original features and features with bilinear upsampling. Fig. 20 shows some examples of up-tiled features. We apply up-tiling with the scale factor of 4 for producing numbers on Tabs 1 and 2 as well as for cluster visualization in Figs 4,5 and Figs 29-34. Unless otherwise stated, no upsampling is performed when computing the cluster assignment.

Code 1: PyTorch code for up-tiling

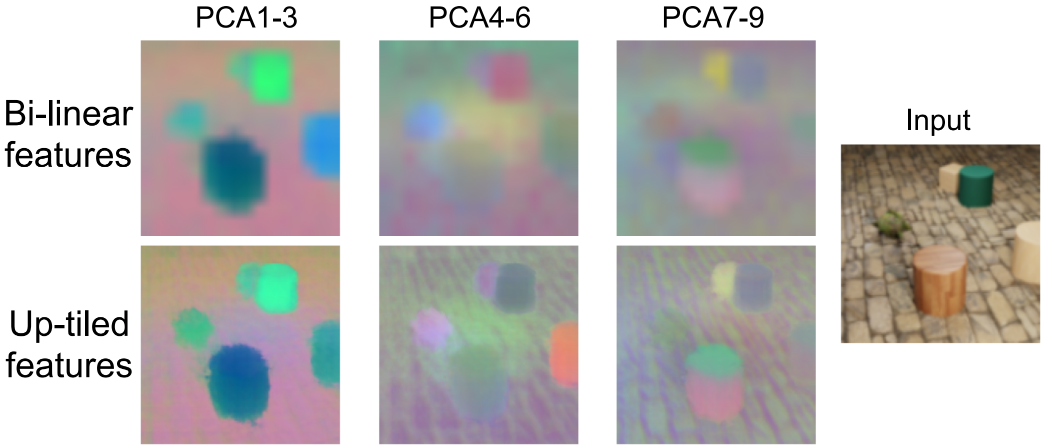
```python
def create_shifted_imgs(img, psize, stride):
    H, W = img.shape[-2:]
    img = F.interpolate(img,
                        (H+psize-stride, W+psize-stride),
                        mode=’bilinear’, align_corners=False)
    imgs = []
    for h in range(0, psize, stride):
        for w in range(0, psize, stride):
            imgs.append(img[:, :, h:h+H, w:w+W])
    return imgs

def uptiling(model, images, psize=16, s=2):
    """
    Args:
        model: a function that takes [B,C,H,W]-shaped tensor
               and outputs [B,C,H/psize,W/psize]-shaped tensor.
        images: a tensor of shape [B, C, H, W].
        psize: the patch size of the model.
        s: scale factor. The resulting features will
            be upscaled to [s*H/psize, s*W/psize]
            where (H, W) are the original image size.
            Must be equal to or less than the patch size.
    Returns:
        nimgs: a tensor of shape [B, C, s*H/psize, s*W/psize]
    """
    B = images.shape[0]
    stride = psize // s
    # Create shifted images.
    shifted_imgs = create_shifted_imgs(images, psize, stride)
    # Compute a feature map on each shifted image.
    outputs = []
    for i in range(len(shifted_imgs)):
        with torch.no_grad():
            output = model(shifted_imgs[i].cuda())
            outputs.append(output.detach().cpu())
    # Tile the output feature maps.
    oh, ow = outputs[0].shape[-2:]
    nimgs = torch.zeros(B, outputs[0].shape[1], oh, s, ow, s)
    for h in range(s):
        for w in range(s):
            nimgs[:, :, :, h, :, w] = outputs[h*s+w]
    # Reshape into [B, C, s*(H/psize), s*(W/psize)]
    nimgs = nimgs.view(, -1, oh*nh, ow*nw)
    return nimgs
```

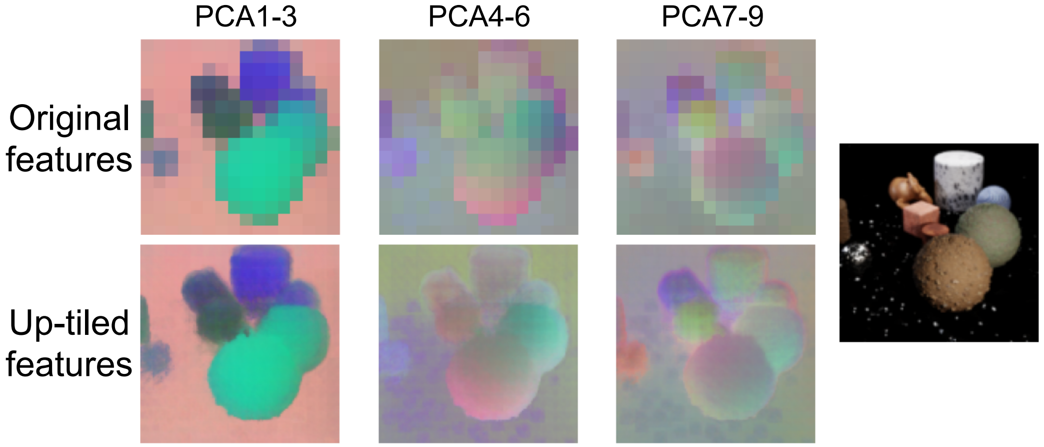
D.2 SUDOKU SOLVING

The task is to fill a 9×9 grid, given some initial digits from 1 to 9, so that each row, column, and 3×3 subgrid contains all digits from 1 to 9. While the task may be straightforward if the game's rules are known, the model must learn these rules solely from the training set. Example boards are shown in Tab. 12.

We train *AKOrN* with attentive connections, the ItrSA model, and a conventional transformer model. We denote them by *AKOrN*$^{\text{attn}}$, ItrSA, and Transformer, respectively. *AKOrN*$^{\text{attn}}$ has almost the same architecture used in the object discovery task except for g in the readout module, which is composed of the norm computation layer followed by a stack of BatchNormalization, ReLU, and linear layer.



(a) The original features (top) and features upsampled by up-tiling (bottom).



(b) Features upsampled by bilinear upsampling (top) and by up-tiling (bottom).

Figure 19: Comparison of *AKOrN*'s output features upsampled by different methods. PCA$\{i - j\}$ indicates that the corresponding column's panels represent the features' i-th to j-th PCA components. The scaling factor of up-tiling is set to 8.

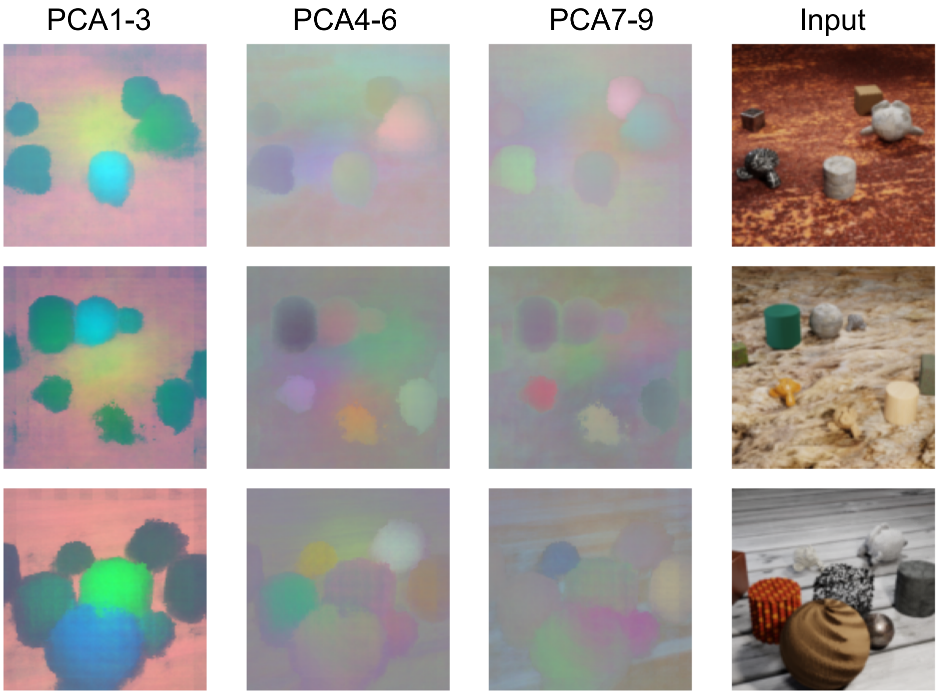
The input for each model is 9×9 digits from 0 to 9 (0 for blank, 1-9 for given digits). We first embed each digit into a 512-dimensional token vector. The 9×9 tokens are then flattened into 81 tokens. We apply each model to this token sequence and compute the prediction on each square by applying the softmax layer to each output token of the final block. All models are trained to minimize cross-entropy loss for 100 epochs.

The number of blocks of both ItrSA and *AKOrN* is set to one. We tested models with more than one block but found no improvement on the ID test set and a decline in OOD performance. Similar to the object discovery experiments, a transformer results in even worse performance than the ItrSA model (Tab. 18).

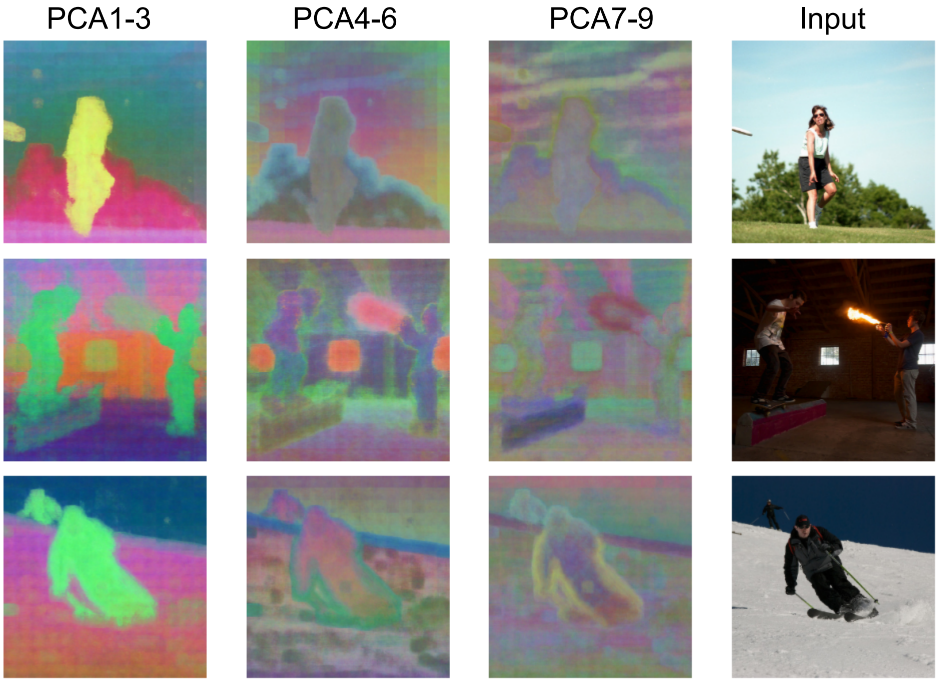
The readout module is composed of the norm computation followed by the Batch Normalization layer, ReLU, and a linear layer.

D.3 ROBUSTNESS AND CALIBRATION ON CIFAR10

We train two types of networks: a convolution-based *AKOrN* and *AKOrN* with a combination of convolution and attention. The former has three proposed blocks, and all of the Kuramoto layer's connectivities are convolutional connectivity. The kernel sizes are 9,7, and 5 from shallow to deep,



(a) CLEVRTex



(b) PascalVOC

Figure 20: Up-tilied feature maps on CLEVRTex and PascalVOC. The scale factors are set to 8 and 16 for CLEVRTex and PascalVOC, respectively.

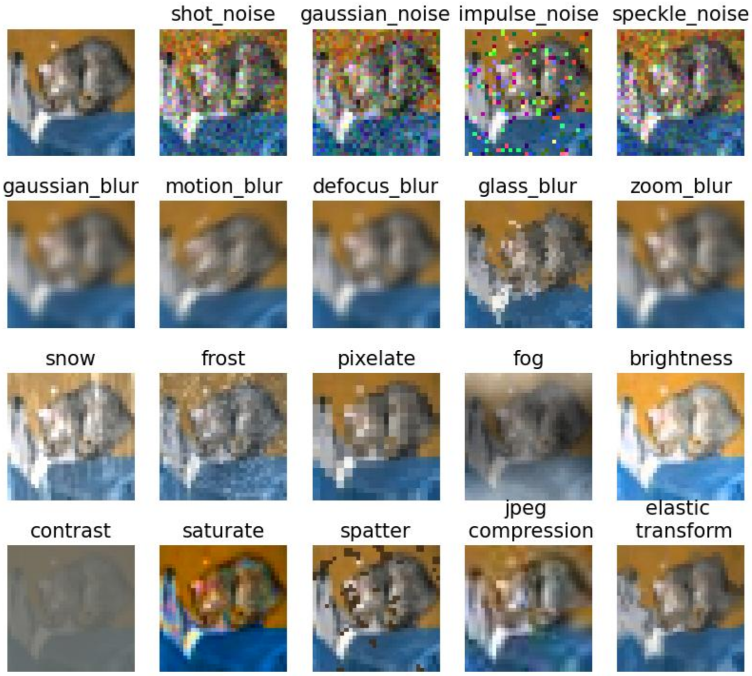


Figure 21: Example images in the Common Corruption dataset (CIFAR10-C). The top right image is the original clean image.

and T is set to 3 for all blocks. Between consecutive blocks, a single convolution with a stride being 2 is applied to each of $\mathbf{C}$ and $\mathbf{X}$. Thus, the feature resolution of the final block's output is 8×8. Each readout module's g is Batch Normalization (Ioffe & Szegedy, 2015) followed by ReLU, and a 3×3 convolution. $\mathbf{C}^{(3)}$ is average-pooled followed by the softmax layer that makes category predictions. The latter network is identical to the former one except for the third block, which we replace with the block with attentive connectivity. For this attentive model, different timesteps T are set across different blocks, which are $[6, 4, 2]$ from shallow to deep.

For ResNet-18 and *AKOrN*, we first conduct pre-training on the Tiny-imagenet (Le & Yang, 2015) dataset with the SimCLR loss for 50 epochs with batchsize of 512. We observe that this pre-training is effective for *AKOrN* and improves the CIFAR10 clean accuracy compared to training from scratch (from 87% to 91%). The ImageNet pretraining slightly improves ResNet's clean accuracy (from 94.1% to 94.4%). Each model is then trained on CIFAR10 for 400 epochs. We apply augmentations, including random scaling and cropping, color jittering, and horizontal flipping, along with AugMix (Hendrycks et al., 2020), as commonly used in robustness benchmarks. Both models are trained to minimize the cross-entropy loss.

We also train an ItrConv model as a non-Kuramoto counterpart for this robustness experiment. To construct the ItrConv model, We replace each block of *AKOrN*$^{\mathrm{conv}}$ with the ItrConv block shown in Fig. 11 and set the same kernel size to each layer as *AKOrN*$^{\mathrm{conv}}$ (i.e. 9, 7, and 5 from shallow to deep layers). Hyperparameters such as the number of channels, learning rate, and others are shared with *AKOrN*$^{\mathrm{conv}}$.

E ADDITIONAL EXPERIMENTAL RESULTS

E.1 POSITIONAL ENCODING FOR THE ATTENTIVE CONNECTIVITY

We need a positional encoding (PE) for *AKOrN* with attentive connectivity. We found GTA-type PE (Miyato et al., 2024) is effective and used for *AKOrN* throughout our experiments. Comparison to absolute positional encoding (APE) (Vaswani et al., 2017) and RoPE (Su et al., 2021) is shown in Tab. 13. GTA does not improve the baseline ItrSA models.

	PE	CLEVRTex FG-ARI	CLEVRTex MBO
ItrSA	APE	66.87	42.15
	GTA	66.07	43.41
AKOrN	APE	71.96	51.35
	RoPE	65.70	50.22
	GTA	75.79	54.08

(a) CLEVRTex

	PE	Sudoku(OOD)
ItrSA	APE	34.37±5.40
	GTA	24.32±7.81
AKOrN	APE	48.13±9.08
	RoPE	48.43±5.60
	GTA	51.72±3.26

(b) Sudoku (OOD Test)

Table 13: Comparison of positional encoding schemes. The number of blocks is one for all models. The Sudoku results of AKOrNs are obtained with test-time extensions of the Kuramoto steps ($T_{\text{eval}} = 128$) but without the energy-based voting.

E.2 UNSUPERVISED OBJECT DISCOVERY

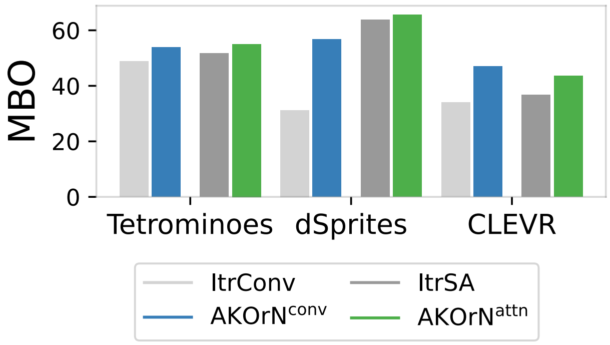
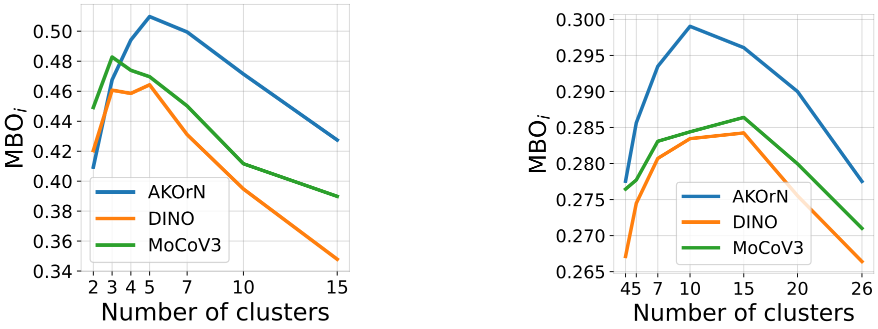


Figure 22: MBO on Tetrominoes, dSprites, and CLEVR.

E.2.1 MBO$_i$ VS # CLUSTERS



(a) PascalVOC

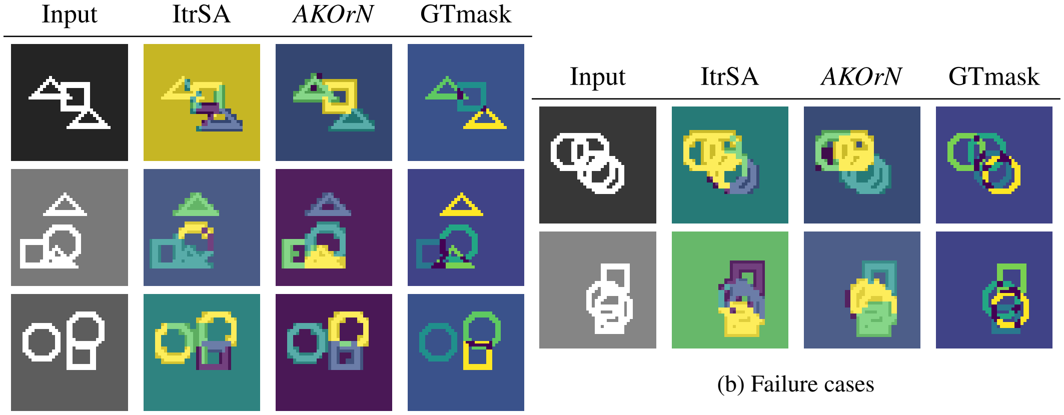
(b) COCO2017

Figure 23: MBO$_i$ vs the number of clusters used for evaluation. *AKOrN* outperforms DINO and MoCoV3 across a wide range of cluster numbers.

E.2.2 OBJECT DISCOVERY PERFORMANCE

Model	Tetrominoes		dSprites		CLEVR	
	FG-ARI	MBO	FG-ARI	MBO	FG-ARI	MBO
ItrConv	55.56	48.82	20.46	31.25	56.41	33.98
AKOrN$^{\text{conv}}$	75.37	53.88	73.35	56.80	65.08	47.03
ItrSA	86.81	51.74	69.42	64.67	80.15	36.76
AKOrN$^{\text{attn}}$	86.19	55.06	79.98	65.57	90.93	43.55
(+up-tiling ($\times$4))						
AKOrN$^{\text{attn}}$	92.72	56.40	88.57	62.81	93.64	43.76
(Distributed representation models)						
CAE (Löwe et al., 2022)	78	-	51	-	27	-
CtCAE (Stanić et al., 2023)	84	-	56	-	54	-
SynCx (Gopalakrishnan et al., 2024)	89	-	82	-	59	-
Rotating Features (Löwe et al., 2023)	42	-	88.8	86.3	66.4	60.8
(Slot-based model)						
Slot-Attnetion (Locatello et al., 2020)	99.5	-	91.3	-	98.8	-

Table 14: Object discovery results on synthetic datasets.



(a) Comparision between ItrSA and *AKOrN*

(b) Failure cases

Figure 24: Cluster visualization on Shapes. (b) Both ItrSA and *AKOrN* sometimes fail at separating overlapping objects with complex configurations.

	L		
	1	2	3
ItrSA	47.5	48.3	49.2
AKOrN$^{\text{attn}}$	**56.2**	**63.6**	**72.6**

(a) FG-ARI

	L		
	1	2	3
ItrSA	38.5	39.0	30.1
AKOrN$^{\text{attn}}$	**41.6**	**45.0**	**48.8**

(b) MBO

Table 15: Object discovery performance on Shapes varying the number of layers L.

Model	CLEVRTex		OOD		CAMO	
	FG-ARI	MBO	FG-ARI	MBO	FG-ARI	MBO
ViT	46.37	23.77	43.60	27.01	31.40	15.75
ItrSA ($L = 1, T = 8,$)	66.07	43.41	65.70	44.50	49.02	29.48
ItrSA ($L = 2, T = 8$)	75.33	48.44	73.91	45.69	60.38	36.72
AKOrN$^{\mathrm{attn}}$ ($L = 1, T = 8$)	75.79	54.94	73.11	55.05	59.70	43.28
AKOrN$^{\mathrm{attn}}$ ($L = 2, T = 8$)	81.50	54.08	80.15	55.02	68.73	44.98
(+up-tiling ($\times 4$))						
AKOrN$^{\mathrm{attn}}$ ($L = 2, T = 8$)	87.28	55.40	86.41	56.32	74.85	45.95
Large *AKOrN*$^{\mathrm{attn}}$ ($L = 2, T = 8$)	89.24	60.02	88.00	60.96	77.18	53.43
*MONet (Burgess et al., 2019)	19.78	-	37.29	-	31.52	-
SLATE (Singh et al., 2022)	44.19	50.88	-	-	-	-
*Slot-Attetion (Locatello et al., 2020)	62.40	-	58.45	-	57.54	-
Slot-diffusion (Wu et al., 2023)	69.66	61.94	-	-	-	-
†SLATE$^+$ (Singh et al., 2022)	70.71	54.90	-	-	-	-
†LSD (Jiang et al., 2023)	76.44	72.44	-	-	-	-
Slot-diffusion+BO (Wu et al., 2023)	78.50	68.68	-	-	-	-
*DTI (Monnier et al., 2021)	79.90	-	73.67	-	72.90	-
*I-SA (Chang et al., 2022)	78.96	-	83.71	-	57.20	-
BO-SA (Jia et al., 2023)	80.47	-	86.50	-	63.71	-
‡NSI (Dedhia & Jha, 2024)	89.89	46.60	-	-	-	-
ISA-TS (Biza et al., 2023)	92.9	-	84.4	-	86.2	-
†Jung et al. (2024)	93.06	75.36	-	-	-	-
^{p}Sauvalle & de La Fortelle (2023)	94.77	-	83.14	-	87.27	-

Table 16: Object discovery on CLEVRTex (Karazija et al., 2021). †Use Openimages (Kuznetsova et al., 2020)-pretrained encoder. Numbers are from Jung et al. (2024). ‡Use ImageNet-pretrained DINO. *Numbers taken from Jia et al. (2023). ^{p}Use Imagenet-pretrained backbone models.

Model	PascalVOC MBO$_i$	PascalVOC MBO$_c$	COCO2017 MBO$_i$	COCO2017 MBO$_c$
(slot-based models)				
Slot-attention (Locatello et al., 2020)	22.2	23.7	24.6	24.9
SLATE (Singh et al., 2021)	35.9	41.5	29.1	33.6
(DINO + synchrony-based models)				
Rotating Features (Löwe et al., 2023)	40.7	46.0	-	-
(DINO + slot-based model)				
NSI (Dedhia & Jha, 2024)	-	-	28.1	32.1
DINOSAUR (Seitzer et al., 2023)	44.0	51.2	31.6	39.7
Slot-diffusion (Wu et al., 2023)	50.4	55.3	31.0	35.0
SPOT (Kakogeorgiou et al., 2024)	48.3	55.6	**35.0**	**44.7**
(SSL models)				
MAE (He et al., 2022)	33.8	37.7	22.9	28.3
DINO (Caron et al., 2021)	44.3	50.0	28.8	35.8
MoCoV3 (Chen et al., 2021)	47.3	53.0	28.7	36.0
AKOrN$^{\text{attn}}$	50.3	58.2	30.2	38.2
(SSL models + up-tiling (×4))				
MAE	34.0	38.3	23.1	28.5
DINO	47.2	53.5	29.4	37.0
MoCoV3	44.6	50.5	29.0	35.9
AKOrN$^{\text{attn}}$	**52.0**	**60.3**	31.3	40.3

Table 17: Object discovery on PascalVOC and COCO2017.

E.2.3 TRAINING EPOCHS VS MBO

Fig. 25 shows that MBO$_i$ and MBO$_c$ scores on Pascal and COCO improve as ImageNet pretraining progresses. Similar observations are made on CLEVRTex datasets, where larger AKOrNs give better object discovery performance (see Figs 30-32 and Tab. 16). These results indicate that there is an alignment between the SSL training with *AKOrN* and learning object-binding features and that increasing parameters and computational resources can further enhance the object discovery performance.

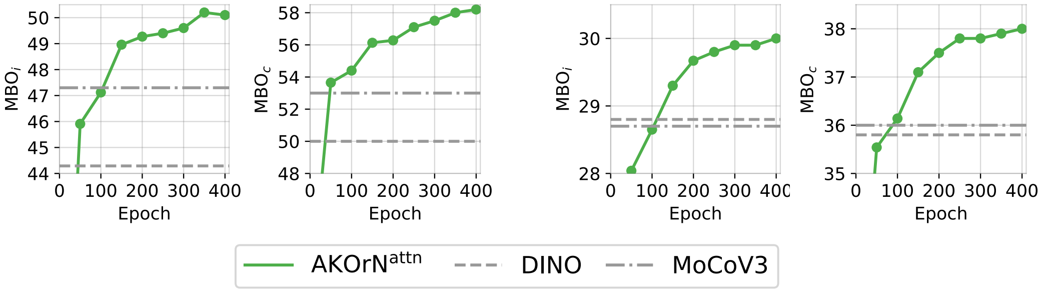


Figure 25: MBO$_i$ and MBO$_c$ vs. training epochs. (Left) PascalVOC (Right) COCO2017.

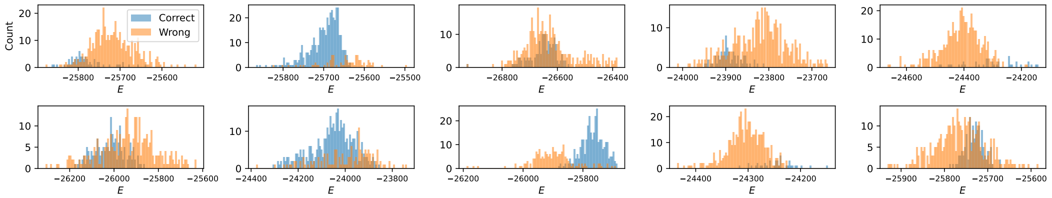
E.3 SUDOKU SOLVING

Model	ID	OOD
Energy Transformer (Hoover et al., 2023)	1.0±1.0	0.0±0.0
Symmetrized *AKOrN* $(L=1, T=16)$	67.7±39.9	1.1±1.7
*Symmetrized *AKOrN* $(L=1, T=16)$	84.6±14.2	1.4±1.7
Transformer	98.6±0.3	5.2±0.3
ItrSA $(L=1, T=16)$	99.7±0.3	14.1±2.7
AKOrN$^{\mathrm{attn}}$ wo $\mathbf{\Omega}$ $(L=1, T=16)$	99.8±0.1	16.6±2.2
AKOrN$^{\mathrm{attn}}$ $(L=1, T=16)$	99.8±0.1	16.6±2.1
(+Test time extensions of internal steps)		
ItrSA $(T_{\mathrm{eval}}=32)$	95.7±8.5	34.4±5.4
AKOrN$^{\mathrm{attn}}$ wo $\mathbf{\Omega}$ $(T_{\mathrm{eval}}=128)$	**100.0**±0.0	49.6±3.3
AKOrN$^{\mathrm{attn}}$ $(T_{\mathrm{eval}}=128)$	**100.0**±0.0	51.7±3.3
$(T_{\mathrm{eval}}=128$, Energy-based voting $(K=100))$		
AKOrN$^{\mathrm{attn}}$ wo $\mathbf{\Omega}$	**100.0**±0.0	46.8±9.0
AKOrN$^{\mathrm{attn}}$	**100.0**±0.0	**61.1**±14.7
SAT-Net (Wang et al., 2019)	98.3	3.2
Diffusion (Du et al., 2024)	66.1	10.3
IREM (Du et al., 2022)	93.5	24.6
RRN (Palm et al., 2018)	99.8	28.6
R-Transformer (Yang et al., 2023)	**100.0**	30.3
IRED (Du et al., 2024)	99.4	**62.1**

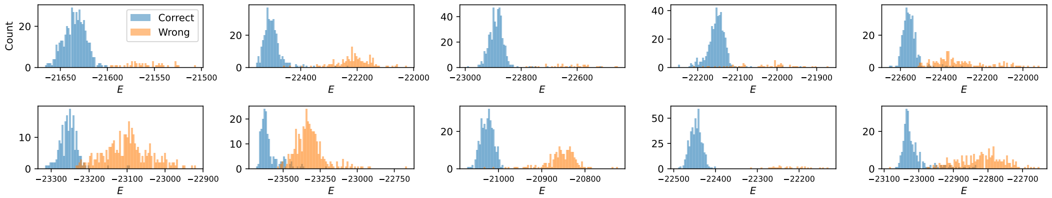
Table 18: Board accuracy on Sudoku Puzzles. The harder dataset (OOD) has fewer conditional digits per example than the train set (17-34 in the harder dataset while 31-42 in the train set). We show the mean and std of the accuracy of models with different random seeds for the weight initialization. *Numbers are calculated with excluding one trained model that has stuck during training.

E.3.1 EFFECT OF THE NATURAL FREQUENCY TERM IN ENERGY-BASED VOTING

Interestingly, the model without the $\mathbf{\Omega}$ term does not give improvement with this post-selection, as the energy value and correctness are inconsistent (Fig. 26). This implies the asymmetric term $\mathbf{\Omega}$ prevents the oscillators from being stuck in bad minima.



(a) without **Ω**



(b) with **Ω**

Figure 26: Energy distribution of the K-Net with or without the **Ω** term. In each panel, given a single board, we compute energies of the final oscillatory states that start from different random oscillators and show the histogram of these energies, color-coded by the correctness of the predictions made on the corresponding final oscillatory states. Note that not for all boards does the model yield those mixed predictions: on approximately 30% boards, all predictions with random initial oscillators are wrong.

E.4 SYMMETRIC CONSTRAINT

Fig. 27 shows a comparison of *AKOrN* to Energy Transformer (Hoover et al., 2023) and a symmetric version of *AKOrN*. The symmetric version *AKOrN* is constructed by using the same weight to compute query and key vectors and a symmetric weight for value vectors. Board accuracies of these symmetrized models are shown in Tab. 18. We observe a similar tendency in the two symmetric models: both models underfit the data (See Fig. 27). Energy Transformer is not able to solve even in-distribution boards. Symmetrized *AKOrN* also gets stuck depending on the seed for the weight initialization.

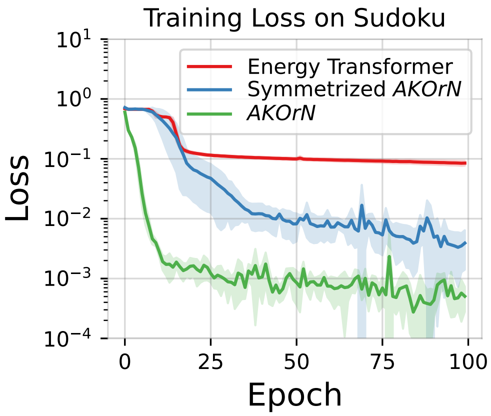


Figure 27: A training curve comparison with symmetric transformer models.

E.5 ROBUSTNESS AND CALIBRATION ON CIFAR10

	$\uparrow$ Accuracy			$\downarrow$ ECE
Model	Clean	Adv	CC	CC
Gowal et al. (2020)	85.29	57.14	69.1	13.2
Gowal et al. (2021)	88.74	66.10	70.7	5.6
Bartoldson et al. (2024)	93.68	73.71	75.9	20.5
Kireev et al. (2022)	94.75	0.00	83.9	9.0
Diffenderfer et al. (2021)	96.56	0.00	89.2	4.8
ViT	91.44	0.00	81.0	9.6
ResNet-18	94.41	0.00	81.5	8.9
ItrConv	93.46	0.00	83.6	5.9
AKOrN$^{\text{conv}}$ ($N = 2$)	88.91	*58.91	83.0	1.3
AKOrN$^{\text{mix}}$ ($N = 2$)	91.23	*51.56	86.4	1.4
AKOrN$^{\text{mix}}$ ($N = 4$)	93.51	*0.00	84.0	6.4

Table 19: (An extended version of Tab. 4) Robustness to adversarial attack (Adv) and Common Corruptions (CC) on CIFAR10 with the most severe corruption level (5). *The adversarial attack is done by AutoAttack with EoT (Athalye et al., 2018). The max norm constraint of the adversrial perturbtions is set to 8/255. With $N = 4$, the performance tendency of *AKOrN* is almost the same as ResNet except for the accuracy and uncertainty calibration on CIFAR10 with natural corruptions, which are moderately better with *AKOrN*$^{\text{mix}}$.

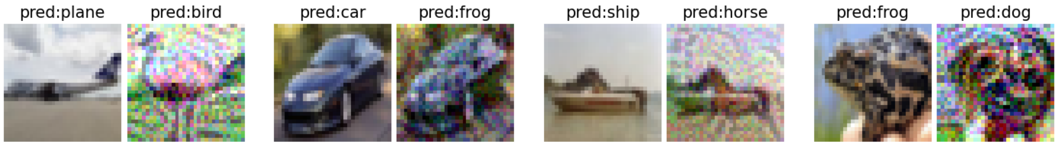
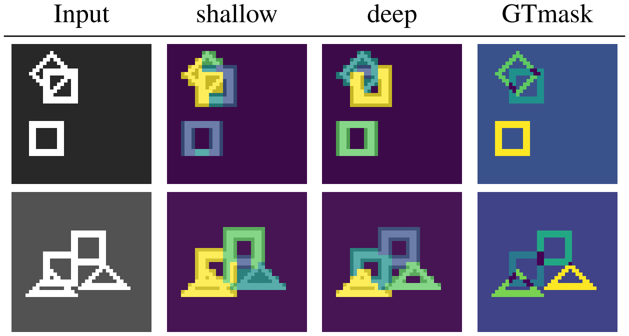
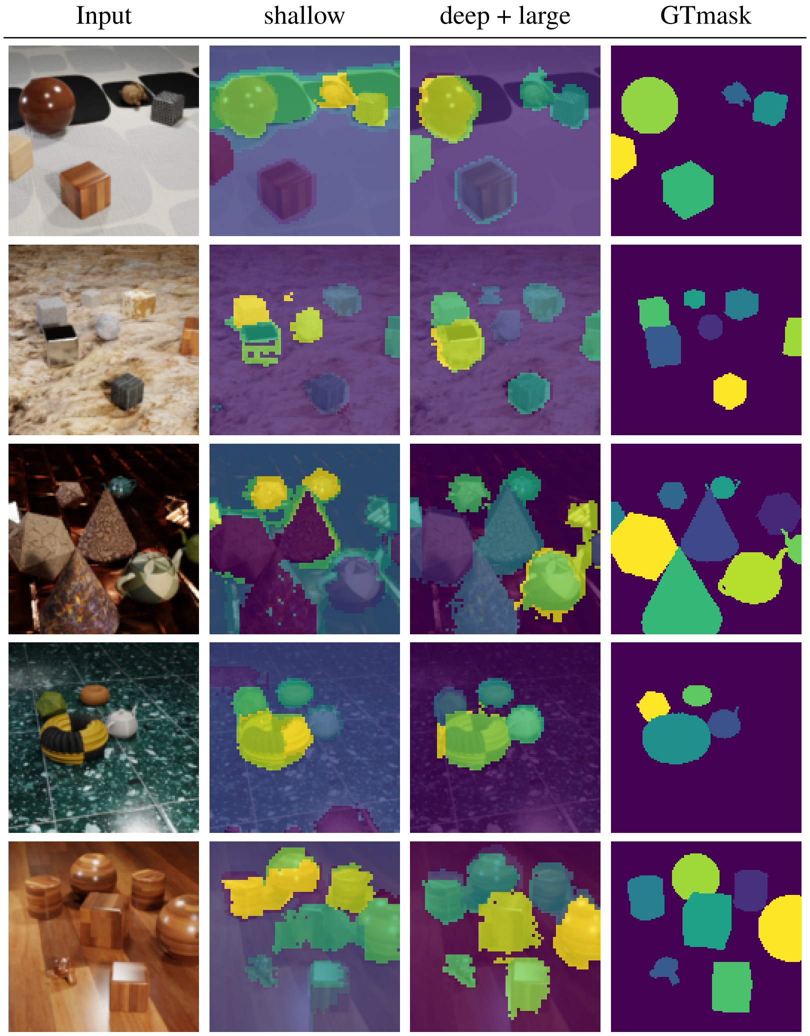


Figure 28: *AKOrN*'s adversarial examples are interpretable. Each pair of images is an original and the adversarially perturbed image ($\|\epsilon\|_\infty = 64/255$). The text above each image indicates the class prediction made by the *AKOrN* model.

F ADDITIONAL CLUSTER VISUALIZATIONS



(a) Shapes



(b) CLEVRTex (1st and 2nd row), CLEVRTex-OOD (3rd and 4th row), and CLEVRTex-CAMO (the last row)

Figure 29: Deeper, wider, and more epochs make the models learn more binding features in *AKOrN*. (a): comparing a single-layer model (shallow) and a 3-layer model (deep). (b): comparing a single-layer model (shallow) and a model with doubled layers, channels, and epochs (deep+large).

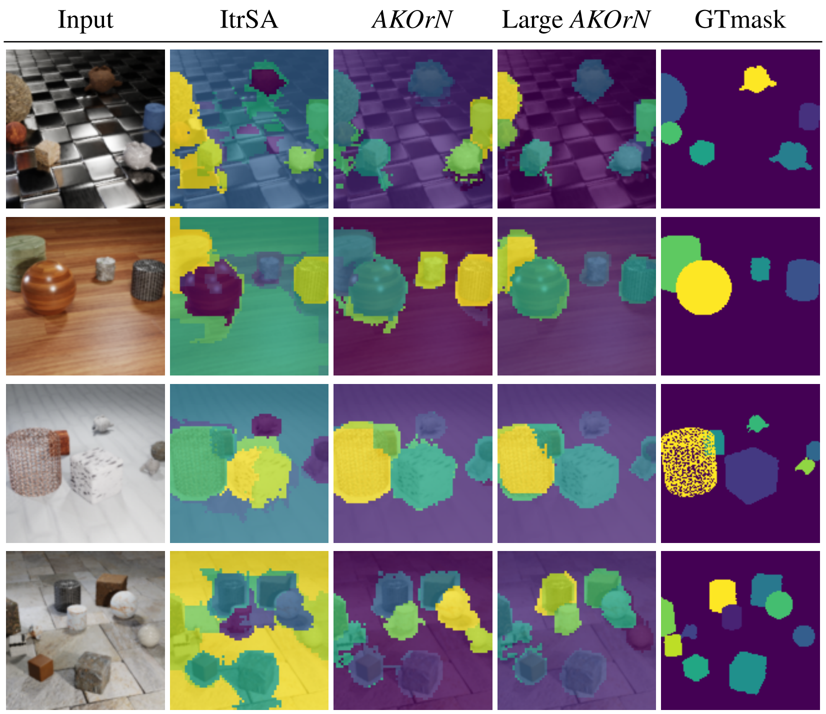


Figure 30: Visualization of clusters on CLEVRTex. The number of blocks L is set to two for all models.

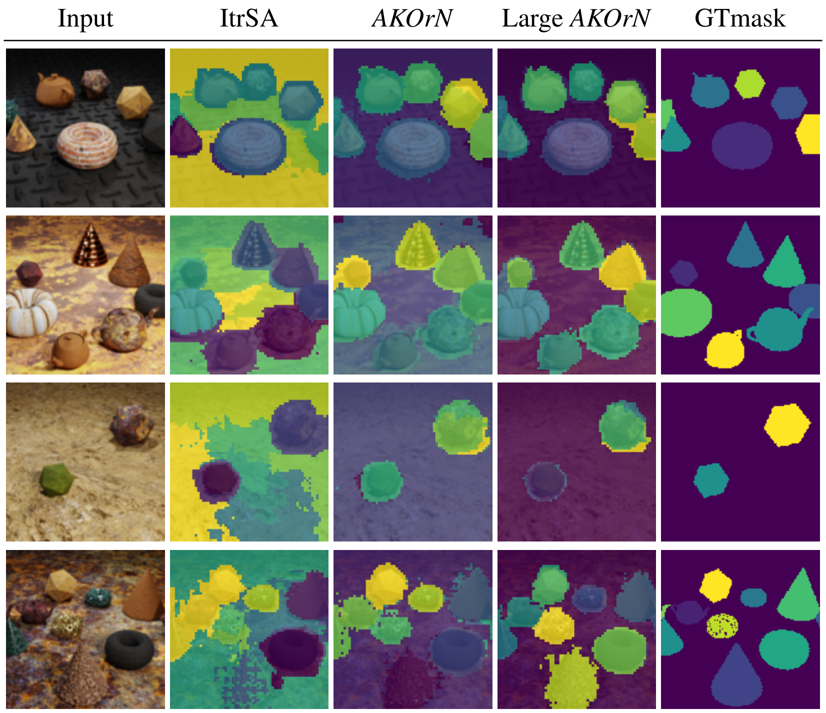


Figure 31: Visualization of clusters on CLEVRTex-OOD. The number of blocks L is set to two for all models.

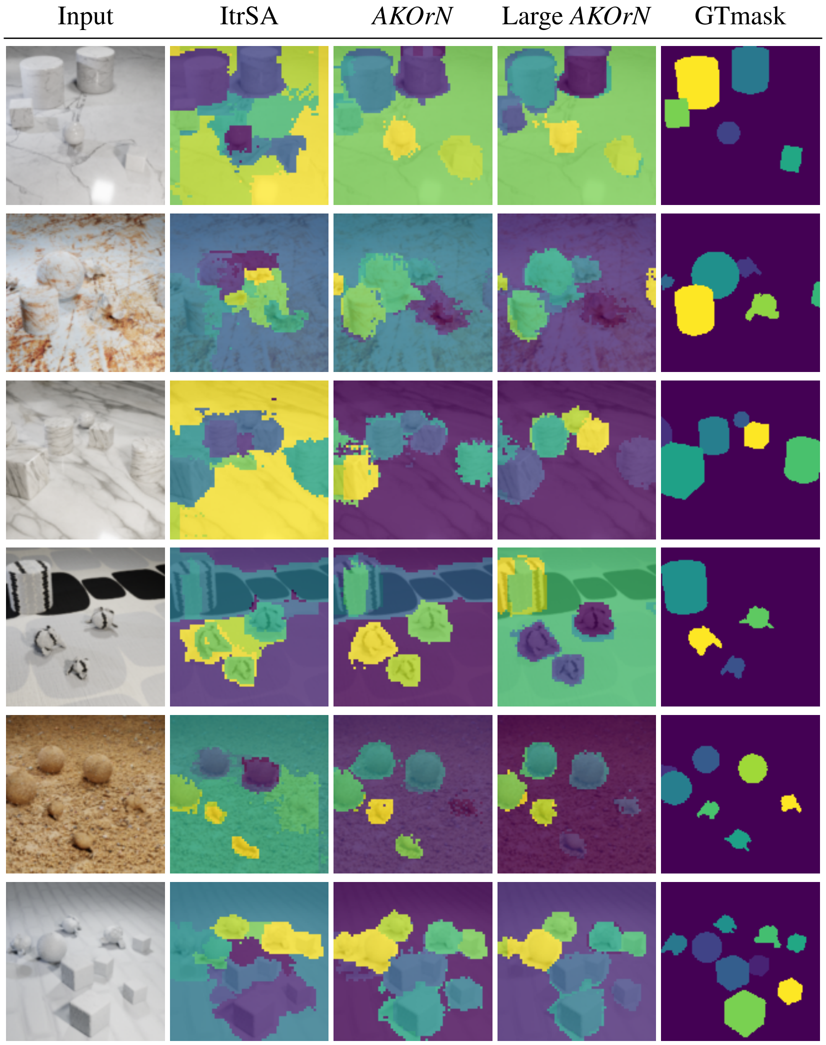


Figure 32: Visualization of clusters on CLEVRTex-CAMO. The number of blocks L is set to two for all models.

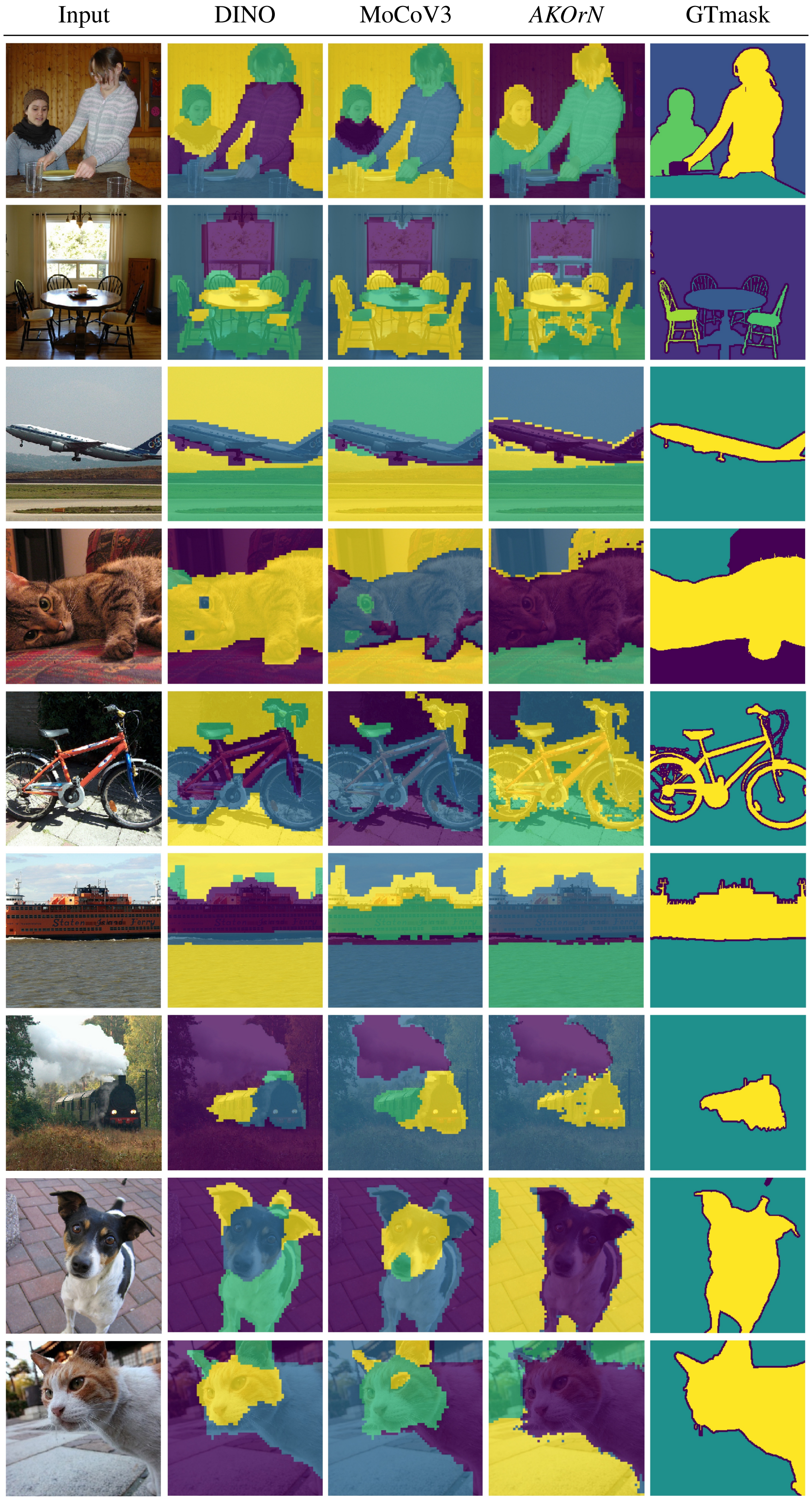


Figure 33: Visualization of clusters on PascalVOC. The number of clusters is set to 4.

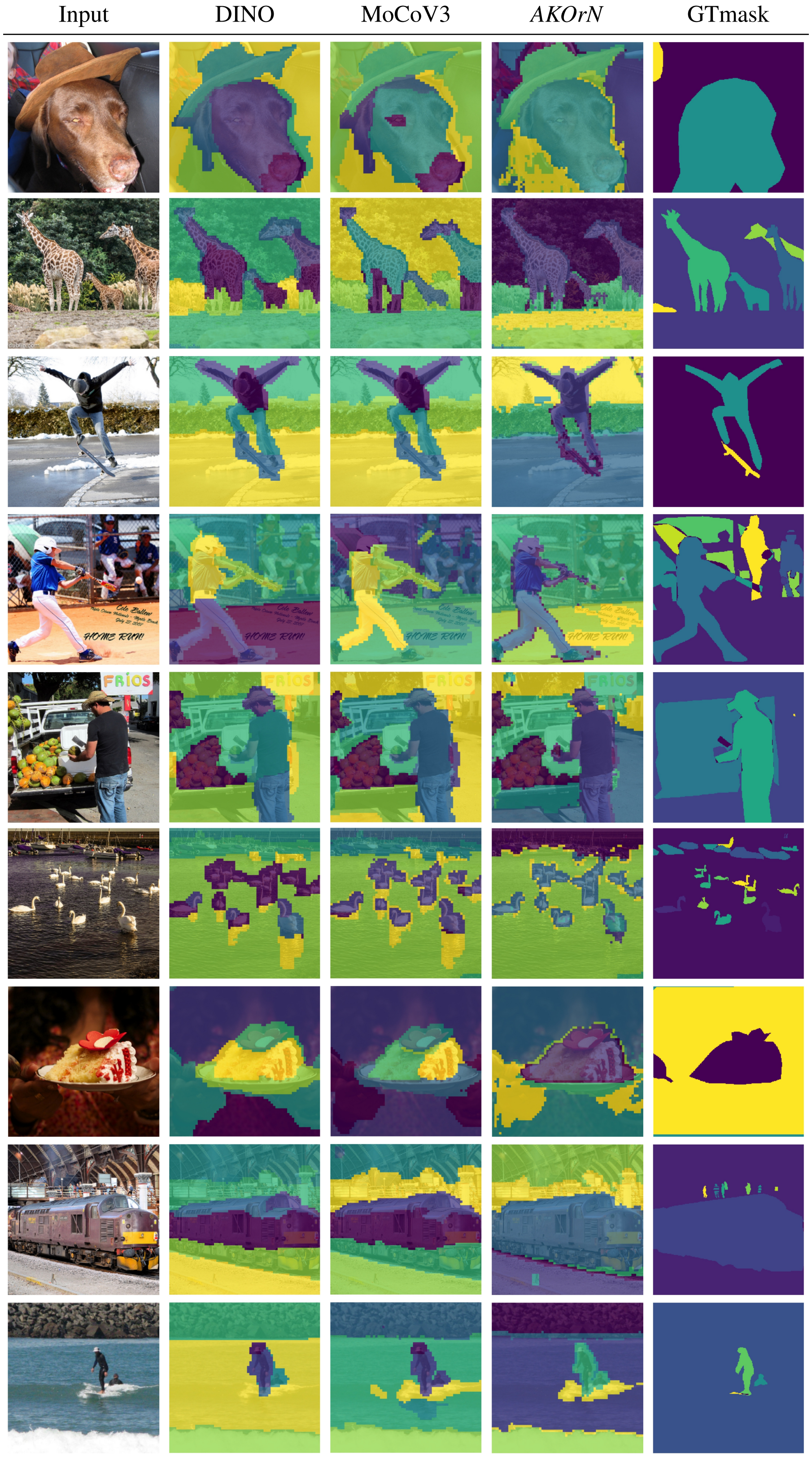


Figure 34: Visualization of clusters on COCO2017. The number of clusters is set to 7.