Infusing Lattice Symmetry Priors in Attention Mechanisms for Sample-Efficient Abstract Geometric Reasoning

Mattia Atzeni [1] **Mrinmaya Sachan** [2] **Andreas Loukas** [3]

Abstract

The Abstraction and Reasoning Corpus (ARC) (Chollet, 2019) and its most recent language-complete instantiation (LARC) has been postulated as an important step towards general AI. Yet, even state-of-the-art machine learning models struggle to achieve meaningful performance on these problems, falling behind non-learning based approaches. We argue that solving these tasks requires extreme generalization that can only be achieved by proper accounting for core knowledge priors. As a step towards this goal, we focus on geometry priors and introduce LATFORMER, a model that incorporates lattice symmetry priors in attention masks. We show that, for any transformation of the hypercubic lattice, there exists a binary attention mask that implements that group action. Hence, our study motivates a modification to the standard attention mechanism, where attention weights are scaled using soft masks generated by a convolutional network. Experiments on synthetic geometric reasoning show that LATFORMER requires 2 orders of magnitude fewer data than standard attention and transformers. Moreover, our results on ARC and LARC tasks that incorporate geometric priors provide preliminary evidence that these complex datasets do not lie out of the reach of deep learning models.

1. Introduction

Infusing inductive biases and knowledge priors in neural networks is regarded as a critical step to improve their sample efficiency (Battaglia et al., 2018; Bengio, 2017; Lake et al., 2017; Lake & Baroni, 2018; Bahdanau et al., 2019). The *Core Knowledge* priors for human intelligence have been studied extensively in developmental science (Spelke & Kinzler, 2007), following the theory that humans are endowed with a small number of separable systems of core knowledge, so that new flexible skills and belief systems can build on these core foundations. Recent research in artificial intelligence (AI) has postulated the idea that the same priors should be incorporated in AI systems (Chollet, 2019), but it is an open question how to incorporate these priors in neural networks.

Following this chain of thought, the Abstraction and Reasoning Corpus (ARC) (Chollet, 2019) was proposed as an AI benchmark built on top of the Core Knowledge priors from developmental science. Chollet (2019) posits that developing a domain-specific approach based on the Core Knowledge priors is a challenging first step and that "*solving this specific subproblem is critical to general AI progress*". Further, he argues that ARC "*cannot be meaningfully approached by current machine learning techniques, including Deep Learning*".

An important category of Core Knowledge priors includes *geometry and topology priors*. Indeed, significant attention has been devoted to incorporating such priors in deep learning architectures by rendering neural networks invariant (or equivariant) to transformations represented through group actions (Bronstein et al., 2021). However, group-invariant learning helps to build models that systematically *ignore* specific transformations applied to the input (such as translations or rotations).

We take a complementary perspective and aim to help neural networks to learn functions that incorporate geometric transformations of their input (rather than to be invariant to such transformations). In particular, we focus on group actions that belong to the symmetry group of a lattice. These transformations are pervasive in machine learning applications, as basic transformations of sequences, images, and other higher-dimensional regular grids fall in this category. While attention and transformers can in principle learn these kind of group actions, we show that they require a significant amount of training data to do so.

To address this sample complexity issue, we introduce LATFORMER, a model that relies on attention masks in order to

[1]EPFL, Switzerland [2]ETH Zurich, Switzerland [3]Prescient Design, Switzerland. Correspondence to: Mattia Atzeni <mattia.atzeni@outlook.it>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

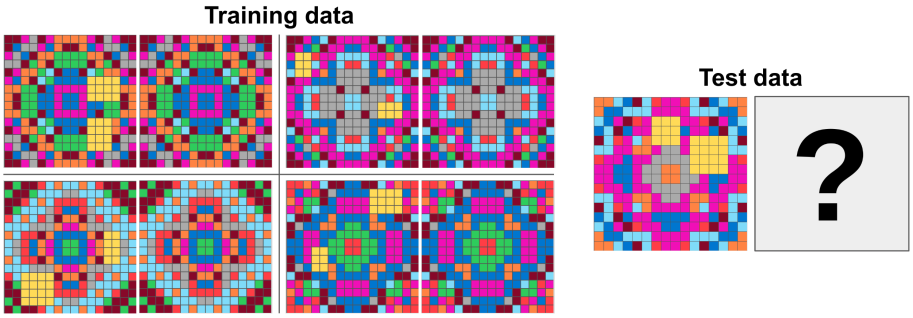


Figure 1: We consider problems that involve learning a geometric transformation on the input data as a sub-problem. The displayed task (taken from ARC) entails learning to map, for each pair, the left to the right image. We investigate how to solve such tasks more sample-efficiently by imbuing self-attention with the ability to exploit lattice symmetry priors.

learn actions belonging to the symmetry group of a lattice, such as translation, rotation, reflection, and scaling, in a *differentiable* manner. We show that, for any such action, there exists an attention mask such that an untrained self-attention mechanism initialized to the identity function performs that action. We further prove that these attention masks can be expressed as convolutions of the identity, which motivates a modification to the standard attention module where the attention weights are modulated by a mask generated by a convolutional neural network (CNN).

Our experiments focus on abstract geometric reasoning and, more specifically, on ARC and its variants, as they are widely regarded as challenging benchmarks for machine learning models (Acquaviva et al., 2021; Chollet, 2019). On these datasets, we aim to reduce the gap between neural networks and hand-engineered search algorithms. To probe the sample efficiency of our method, we compared its ability to learn synthetic geometric transformations against Transformers and attention modules. Then, we annotated ARC tasks based on the knowledge priors they require, and we evaluated LATFORMER on the ARC tasks requiring geometric knowledge priors. Finally, we performed experiments on the more recent *Language-complete ARC* (LARC) (Acquaviva et al., 2021), which enriches ARC tasks with natural-language descriptions, and we compared our model against strong baselines based on neural program synthesis. Our results provide evidence that LATFORMER can learn geometric transformations with 2 orders of magnitude fewer training data than transformers and attention. We also significantly reduce the gap between neural and classical approaches on ARC, providing the first neural network that reaches good performance on ARC tasks requiring geometric knowledge priors.

2. Formalizing the Group-Action Learning Problem

To build intuition on the kind of basic priors that we aim to infuse in neural networks, Figure 1 shows a task borrowed from ARC (Chollet, 2019). The task entails learning to fill out the yellow patches in the leftmost image (input) so that the resulting image satisfies a 90° rotation symmetry. The learner is given only a small set of input-output pairs: the ARC tasks have 3.3 training examples on average. Though the task is challenging for a general neural network (due to the small number of examples), it becomes easier under the prior knowledge of discrete two-dimensional point groups, one of which is the cyclic group of 4-fold rotations C_4. Under this prior, it can be solved by the composition of a group action (rotating each image x by some $\mathsf{g} \in \mathsf{C}_4$) and a shallow neural network with a non-linear activation (mapping yellow to zero and taking a pixel-wise max).

More formally, we are interested in helping neural networks learn lattice transformations in a sample efficient manner by infusing knowledge priors in the model. Motivated by ARC, we focus on learning geometric transformations that belong to the symmetry group of a lattice. This pertains to the more general problem of learning group actions given the input and the output of the transformation (group-action learning).

Concretely, we consider input-output transformations involving a group element g taken from some known group G that can be expressed under the general formulation:

$$y = f(\mathsf{g} \circ x, x), \quad \mathsf{g} = g(x) \in \mathsf{G}. \quad \text{(group-act. learning)}$$

Above, $x \in \mathbb{R}^{d_{\text{in}}}$ and $y \in \mathbb{R}^{d_{\text{out}}}$ are input and output examples, f, g are unknown functions, and $\circ$ denotes the application of a group action. As seen, the group element g can depend on the input data itself. More generally, the function f may depend on more than one transformations of x based on elements belonging to various groups of interest.

It is important to stress that group-action learning is the exact antithesis of the typical group invariant and equivariant learning problems (Bronstein et al., 2021):

$$y = f(\mathsf{g} \circ x) \quad \text{for } \textit{every } \mathsf{g} \in \mathsf{G} \quad \text{(invariant learning)}$$
$$\mathsf{g} \circ y = f(\mathsf{g} \circ x) \quad \text{for } \textit{every } \mathsf{g} \in \mathsf{G}. \quad \text{(equiv. learning)}$$

Intuitively speaking, whereas in group-action learning one aims to learn functions that involve specific (and data-dependent) transformations of our data by actions of the group, in in/equivariant learning the goal is to build models that are oblivious to such actions in a systematic manner.

3. Attention Masks for Core Geometry Priors

This section prepares some theoretical grounding for LATFORMER, our approach to learn the transformations for lattice symmetry groups in the form of attention masks. The section defines attention masks and explains how they can be leveraged to incorporate geometry priors when solving group action learning problems on sequences and images.

3.1. Modulating Attention Weights with Soft Masking

Consider the scaled dot-product attention mechanism as defined in Vaswani et al. (2017). In our formulation, we consider real-valued masks $\boldsymbol{M} \in [0,1]^{n_Q \times n_K}$ that rescale attention weights:

$$\boldsymbol{A} = \mathrm{softmax}\Big(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}}\Big) \odot \boldsymbol{M},$$

where $\boldsymbol{Q} \in \mathbb{R}^{n_Q \times d}$ is the query parameter of the attention mechanism, $\boldsymbol{K} \in \mathbb{R}^{n_K \times d}$ is the key, d is the dimensionality of the model, n_Q and n_K are the sizes of the sets encoded by the query and key matrices respectively, and $\odot$ is the Hadamard product. Attention masks have been widely used to constrain the values of the attention weights and are usually binary masks applied before the $\mathrm{softmax}$ activation (Vaswani et al., 2017; Sartran et al., 2022). However, as we aim to learn $\boldsymbol{M}$, we apply the mask after the $\mathrm{softmax}$ operation in order to avoid squashing the gradient. Therefore, we rescale the attention weights to sum to 1 when calculating the output $\boldsymbol{X}$ of the attention mechanism:

$$\mathrm{MaskedAttention}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V};\boldsymbol{M}) = \frac{\boldsymbol{A}}{\boldsymbol{A}\cdot \mathbf{1}_{n_K}\mathbf{1}_{n_K}^\top}\boldsymbol{V},$$

with $\mathbf{1}_n$ being a vector of ones of size n and $\boldsymbol{V} \in \mathbb{R}^{d \times n_K}$ being the value parameter of the attention mechanism. Though masking can also be applied in cross-attention, in the following we primarily focus on *self-attention*, where $\boldsymbol{Q} = \boldsymbol{K} = \boldsymbol{V} = \boldsymbol{X}$. For ease of notation, we write $\mathrm{MaskedAttention}(\boldsymbol{X};\boldsymbol{M})$ whenever the query, key and value are the same matrix $\boldsymbol{X}$.

3.2. Existence of Attention Masks Implementing Lattice Symmetry Actions

This section discusses group actions that can be represented by attention masks. To develop intuition, let us first consider the simple example of translation in a one-dimensional lattice. Supposing that $\boldsymbol{x} = (x_1,\ldots,x_n)^\top$ is a vector of n elements, we have:

$$\mathrm{MaskedAttention}(\boldsymbol{x};\boldsymbol{M}) = (x_n, x_1, \ldots, x_{n-1})^\top$$

with:

$$\boldsymbol{M} = \begin{pmatrix} 0 & 0 & \cdots & 1 \\ 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 \end{pmatrix}.$$

Hence, when $\boldsymbol{M}$ is the circulant permutation matrix shown above, we have that the mask shifts the input $\boldsymbol{x}$ by one element to the right.

Beyond translation, it is natural to ask what kinds of group actions we can perform with attention masks on data with a more high-dimensional topological structure. The following theorem provides existence statements for data whose underlying topological space is a hypercubic lattice (such as sequences, images and higher-dimensional regular grids).

Theorem 3.1 (Existence)**.** *Let G_m be the symmetry group of the m-dimensional hypercubic lattice, including translational symmetry, 4-fold rotational symmetry and vertical, horizontal and diagonal reflections. Let $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ be a vectorized representation of an m-dimensional tensor $\mathbf{X} \in \mathbb{R}^{l_1\times\cdots\times l_m}$, with $n = l_1 \cdot \ldots \cdot l_m$. For any group action $\mathsf{g} \in \mathsf{G}_m$, there exists an attention mask $\boldsymbol{M}_\mathsf{g} \in \{0,1\}^{n\times n}$, such that:*

$$\mathrm{MaskedAttention}(\boldsymbol{X};\boldsymbol{M}_\mathsf{g}) = \mathsf{g} \circ \boldsymbol{X}.$$

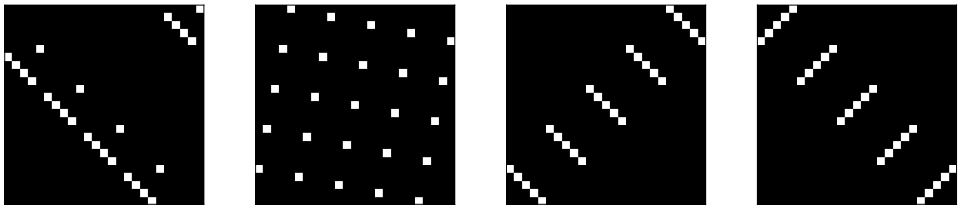
In other words, Theorem 3.1 states that any translation, rotation or reflection can be expressed in terms of an attention mask. Figure 2 shows some examples of masks corresponding to translation, rotation and reflection operations on square lattices.

In the following, we adopt the convention of writing $\boldsymbol{M}_\mathsf{g}$ to mean the mask that implements action g. For more details and for a proof of Theorem 3.1, we refer the reader to Appendix D.

3.3. Representing Attention Masks for Lattice Transformations

To facilitate the learning of lattice symmetries, one needs to determine methods to parameterize the set of feasible group elements. Fortunately, as precised in the following theorem, the attention masks considered in Theorem 3.1 can be expressed conveniently under the same general formulation.

Theorem 3.2 (Representation)**.** *Let G_m be the symmetry group of the m-dimensional hypercubic lattice and $\mathsf{g} \in \mathsf{G}_m$ be an action on a tensor $\mathbf{X} \in \mathbb{R}^{l_1\times\cdots\times l_m}$. Then, there exist some primitive attention masks $\boldsymbol{M}_{\mathsf{g}_i} \in \{0,1\}^{n_i\times n_i}$ such*



(a) Translation (1, 1) (b) Rotation by 90° (c) Vertical reflection (d) Horiz. reflection

Figure 2: Examples of attention masks implementing transformations in two dimensions, including: (a) translation by 1 pixel on both axes, (b) rotation by 90° counterclockwise, (c) vertical reflection and (d) horizontal reflection around the center. White represents value 1 and black 0.

Table 1: Fourier shifts for the transformations on the 1-dimensional and square lattices. We denote with $\boldsymbol{o}(\mathsf{g}_i)_k$ the k-th component of the vector $\boldsymbol{o}(\mathsf{g}_i) \in \mathbb{R}^n$, for $k = 1, \ldots, n$. As stated in Theorem 3.2, attention masks for higher-dimensional lattices can be obtained by the Kronecker product of primitive masks defined over the 1-dimensional and square lattices. Composition of actions is given by matrix multiplication of the masks.

Transformation	**Fourier shift**	**Size of X**
Identity	$\boldsymbol{o}(\mathsf{g}_i)_k = 0$	$n = l_1$
Translation (by δ)	$\boldsymbol{o}(\mathsf{g}_i)_k = -\delta$	$n = l_1$
Reflection	$\boldsymbol{o}(\mathsf{g}_i)_1 = (n-1)$, $\boldsymbol{o}(\mathsf{g})_k = \boldsymbol{o}(\mathsf{g})_{k-1} - 2$	$n = l_1$
Rotation ($90°$)	$\boldsymbol{o}(\mathsf{g}_i)_k = k \cdot (l_1 - 1) - \lfloor (k-1)/l_1 \rfloor$	$n = l_1 \times l_2$
Upscaling (by h)	$\boldsymbol{o}(\mathsf{g}_i)_k = (k - 1 \bmod h) + (h-1) \cdot \lfloor (k-1)/h \rfloor$	$n = l_1$

that

$$\boldsymbol{M}_{\mathsf{g}} = \bigotimes_i \boldsymbol{M}_{\mathsf{g}_i} \quad \textit{and}$$

$$\mathcal{F}(\boldsymbol{M}_{\mathsf{g}_i}) = \mathcal{F}(\boldsymbol{I}_{n_i}) \, \exp(-\frac{2\pi j}{n_i} \, \boldsymbol{o}(\mathsf{g}_i) \, \boldsymbol{r}_{n_i}^\top),$$

where $\boldsymbol{M}_{\mathsf{g}} \in \{0,1\}^{n \times n}$ is an attention mask implementing g, $\mathsf{g}_i \in \mathsf{G}_{m_i}$ for some $m_i \in \{1, 2\}$ is an action on the one-dimensional or square lattice, $\otimes$ is the Kronecker product, $\mathcal{F}$ is the Fourier transform applied column-wise, $\boldsymbol{I}_{n_i}$ is the $n_i \times n_i$ identity matrix, j is the imaginary unit, $\boldsymbol{r}_{n_i} = (1, 2, \ldots, n_i)^\top$, and $\boldsymbol{o}(\mathsf{g}_i)$ is defined as in Table 1.

To obtain an intuitive understanding of Theorem 3.2, it helps to revisit the example of translation by $\delta = 1$ of a sequence $\boldsymbol{x} \in \mathbb{R}^n$ on the 1-dimensional lattice ($m = 1$). Consulting Table 1, we find that $\boldsymbol{o}(\mathsf{g})$ is a vector containing -1 at every position and we know $\boldsymbol{M}_{\mathsf{g}}$ is the permutation circulant matrix of Section 3.2. Indeed, by the time-shifting property of the Fourier transform, $\boldsymbol{M}_{\mathsf{g}}$ can be obtained by shifting the rows of the identity by -1. In general, vector $\boldsymbol{o}(\mathsf{g})$ has a convenient intuitive interpretation as its k-th component represents the relative position (with respect to k) of the element that the k-th row of $\boldsymbol{X}$ attends to. For instance, in the one-dimensional example of translation by one element to the right, each element attends to the one immediately before. Hence, we have $\boldsymbol{o}(\mathsf{g})_k = -1$ for any $k = 1, \ldots, n$.

For higher-dimensional lattices, attention masks can be expressed as the Kronecker product of the attention masks for lower-dimensional cases. For instance, on the square lattice, a translation by 1 pixel on both dimensions is the Kronecker product of the two circulant matrices corresponding to a translation by 1 pixel on the one-dimensional lattice, as shown in Figure 2a. On more than one dimension, we can additionally define 4-fold rotations, still following the same formulation, with $\boldsymbol{o}(\mathsf{g}_i)$ defined as in Table 1.

Although strictly not a symmetry operation, *scaling transformations* of the lattice can also be defined in terms of attention masks under the same general formulation of Theorem 3.2, as reported in Table 1. Therefore, for completeness, we will consider scaling transformations as well in our experiments.

Notice that Theorem 3.2 allows us to derive a way to calculate the attention masks. In particular, we can express our attention masks as a convolution operation on the identity, as stated below.

Corollary 3.3. *Let G_m be the symmetry group of the m-dimensional hypercubic lattice and let $\boldsymbol{M}_{\mathsf{g}} \in \mathbb{R}^{n \times n}$ be an attention mask implementing action $\mathsf{g} \in \mathsf{G}_m$. Then:*

$$\boldsymbol{M}_{\mathsf{g}}[:, i] = \mathcal{F}^{-1}(\exp(-\frac{2\pi j}{n} \cdot \boldsymbol{o}(\mathsf{g}) \cdot \boldsymbol{r}_n^\top))[:, i] \star \boldsymbol{I}_n[:, i],$$

where $\star$ denotes the convolution operation.

In other words, we can represent any mask in our framework as a convolution of the identity matrix with predefined kernels. This motivates us to design a convolutional neural network that produces our attention masks by successive convolutions of the identity.

4. The LATFORMER Architecture

While in principle the problem of inferring group actions from input-output pairs can be solved via search over finite groups, in practice the size of the group for lattice symmetry actions makes this approach unfeasible[1]. Moreover, we are interested in learning unknown functions jointly with the transformation, which cannot be solved by searching on the space of group actions. Using a neural agent to search the space of possible actions would be a viable alternative, but this would make the problem non-differentiable and we would need to resort to reinforcement learning methods.

In this work, we aim to solve the problem in a *differentiable* way. Inspired by the observations above, we introduce LATFORMER, which incorporates the insights of Section 3 into a neural architecture. We propose to use gated CNNs to parameterize the masks and we introduce an additional smoothing technique for easier optimization.

4.1. Lattice Mask Experts as Convolutional Networks

Attention modules in neural networks usually include an attention mechanism with learnable linear transformations of the inputs[2] followed by a feed-forward network (FFN), as in the Transformer encoder layer (Vaswani et al., 2017).

To infuse core geometry priors in the attention module, we propose to modulate the attention weights with a mask generated by an additional layer, as shown in Figure 3a. We refer to this layer as *Lattice mask expert*, as it specializes towards specific transformations of the lattice. To understand the purpose of this layer, it is useful to remember that, by the analysis conducted in Section 3, even if the attention and FFN layers are initialized to the identity function, the mask expert can generate attention masks that produce precise geometric transformations of the input.

By Corollary 3.3, we know that each group action on the lattice can be represented by a mask that is a convolution of the identity and we have an analytical expression to calculate the kernels of the convolution. We can leverage this notion to design CNNs that produce attention masks corresponding to specific group actions by following the general formulation:

$$\begin{aligned} \boldsymbol{M}_0 &= \boldsymbol{I}, \\ \boldsymbol{M}_{l+1} &= \alpha_l \,\mathrm{Conv}(\boldsymbol{M}_l, \boldsymbol{K}_l) + (1-\alpha_l)\boldsymbol{M}_l \end{aligned}$$

for $l = 0, \ldots, L-1$. Above, $\boldsymbol{M}_L$ is the predicted mask, $\alpha_l = \sigma_l(\boldsymbol{X}; \boldsymbol{\theta}) = \mathrm{FFN}_l(\boldsymbol{X}, \boldsymbol{\theta})$ is the output of a gating function, $\boldsymbol{\theta}$ is a learnable parameter, and $\boldsymbol{K}_l$ is the kernel of the l-th convolutional layer whose weights are determined based on Corollary 3.3 and Table 1.

As an example, Figure 3b shows an architecture that generates translation masks. Following Theorem 3.2, the expert computes the translations along the two dimensions separately and then aggregates the resulting masks doing the Kronecker product. Hence, a *Lattice translation expert* with L convolutional layers for each dimension can generate any translation up to $\delta = 2^L - 1$ elements per dimension. At inference time, the values of the gates can be discretized, in such a way that the generated mask provably performs a meaningful group action.

Similarly to the expert in Figure 3b, we can define gated CNNs for rotation, reflection, and scaling. The product of experts (i.e., the combination of more actions) can be obtained by either chaining the experts or multiplying the attention masks generated by different experts. For more details, we refer the reader to Appendix A.

4.2. Mask Smoothing for Easier Training

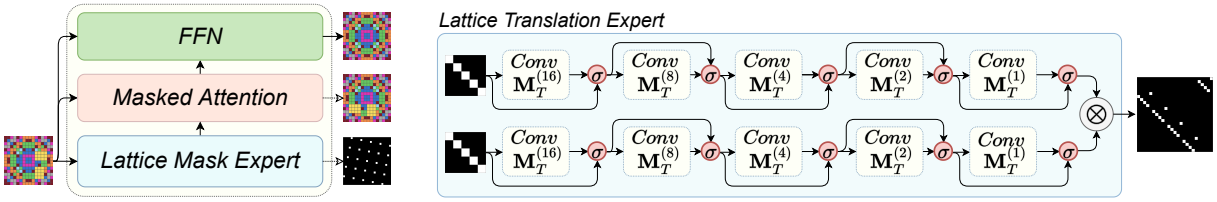
The framework described so far parameterizes discrete transformations of a lattice in a differentiable manner. Nevertheless, to improve the training of LATFORMER, we found it beneficial to also apply a smoothing operation on the attention masks. Our approach entails defining an adjacency relation between group elements and applying graph convolution with a heat kernel on the corresponding graph. This encourages the optimizer to favor weight updates that change the masks in a smooth manner w.r.t. the geodesic distance implied by the graph. Concretely, we define the neighbors of each element g_i on the lattice as those $\mathsf{g}_j = \mathsf{e} \circ \mathsf{g}_i$ reachable by an application of a primitive action e, such as translation by a single pixel in one dimension, rotation by $90°$, and vertical/horizontal reflection. The notion of neighborhood gives rise to a graph whose vertex set is the lattice group and that contains one edge for every pair of neighboring actions.

As before, it helps to consider different kinds of transformations separately. For instance, as shown in Figure 4, for 2D rotations the underlying graph is a cycle with 4 elements due to the underlying point group for 4-fold rotations being the cyclic group C_4. Performing heat diffusion can be achieved by repeated neighborhood averaging over the cycle and yields a smoothed rotation mask that performs all rotations at the same time (rightmost image in Figure 4). We can extend the same approach to all lattice transformations: for instance, in the case of translation, the underlying graph is a grid and the smoothing operation is akin to convolution with a Gaussian kernel.

To train LATFORMER with smoothed masks, we compute two predictions: one with the non-smooth mask predicted by the model and one with a smoothed version of the same

[1] The size of the groups we consider grows with a polynomial of n and exponentially with m.

[2] For simplicity, we omitted linear transformations in the definition of $\mathrm{MaskedAttention}$ in Section 3.1.



(a) (b)

Figure 3: A LATFORMER layer (a) and an architecture for a *Lattice translation expert* (b). The LATFORMER layer (a) is a standard Transformer encoder layer augmented with a *Lattice mask expert* constrained to generate attention masks corresponding to a geometric transformation of the input. The *Lattice translation expert* (b) is a particular instance of a *Lattice mask expert* that produces translation masks. In the architecture above, every convolutional layer is meant to shift the input by a power of 2 and can be skipped by a gating function (denoted as σ).

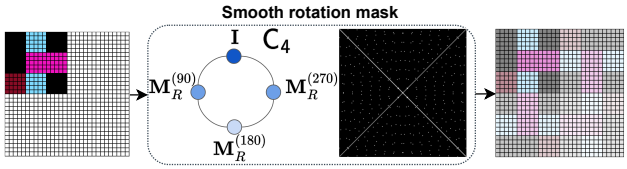


Figure 4: Rotational smoothing can be obtained by heat diffusion over the cyclic graph of rotation masks.

mask. The final loss is the sum of two cross-entropy losses calculated separately for the two predictions.

5. Experiments

To evaluate our method, we first developed a set of synthetic tasks in order to compare LATFORMER to attention modules and Transformers with respect to sample efficiency in learning basic geometric transformations. Then, we annotated the ARC tasks based on the knowledge priors they require, and we assessed the performance of our method on this challenging dataset. Finally, we experimented with the LARC (Acquaviva et al., 2021) dataset and compared our method to stronger baselines based on neural program synthesis. We report additional experimental results in Appendix B.5.

5.1. Sample Efficiency on Geometric Transformations

As a preliminary study, we probed the ability of LATFORMER to learn geometric transformations efficiently. To this end, we compared the performance of our model to a transformer (Vaswani et al., 2017) and an attention module (the same architecture as our approach, without the mask expert) on synthetic tasks with increasing number of examples. Inspired by ARC, we generated a set tasks where the model needs to infer a geometric transformation from input-output pairs. The input is a grid taken from the ARC tasks and the output is either a translation, rotation, reflection or scaling of the input. The specific transformation applied to the input grid defines the task and is consistent across all examples in the same task.

We evaluated the models based on the mean accuracy across tasks. Figure 5 shows the accuracy of our model compared to the baselines and to a version of LATFORMER without smoothing. The plots show that LATFORMER can generalize better and from fewer examples than transformers and attention modules both with absolute positional encodings (Vaswani et al., 2017) and relative positional encodings (Shaw et al., 2018). Additionally, our results show that the smoothing operation described in Section 4.2 is helpful for larger groups. More details on this experiment are reported in Appendix B.1.

5.2. Geometric Reasoning on ARC Tasks

To assess the ability of our approach to learn efficiently on a more challenging use case, we focused on a subset of the ARC dataset (Chollet, 2019) requiring geometric priors for which our method could be a principled solution. To this end, we annotated the ARC tasks based on the knowledge priors they require, using the list of priors provided by Chollet (2019) as a reference. Appendix B.2 provides more details about the annotation of ARC and Figure 7a in the Appendix shows the knowledge priors that we considered and their distribution across the ARC tasks.

We assessed the performance of our model on the tasks that require only knowledge priors corresponding to the basic geometrical transformations that we addressed in this work, namely translation, rotation, reflection and scaling. Table 2 shows our results compared to neural baselines, including CNNs, attention with relative positional encodings (Shaw et al., 2018), PixelCNN (Gul et al., 2019), and Transformers (Vaswani et al., 2017), and a Differentiable Neural Computer (Graves et al., 2016) with spectral regularization (Kolev et al., 2020). We additionally compared to a Transformer model that has access to precomputed transformations of the input (**Transformer + data augmentation**). Precomputing

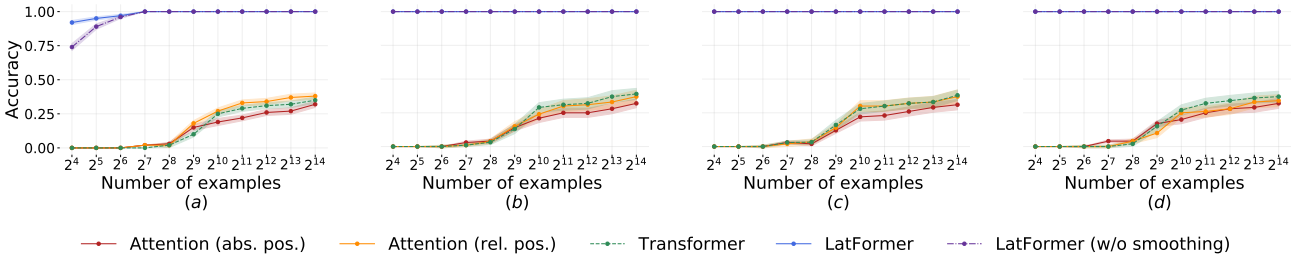


Figure 5: Sample efficiency of our method compared to the baselines on synthetic tasks on translation (a), rotation (b), reflection (c) and scaling (d). The y axis denotes the mean accuracy across tasks belonging to the same category, whereas the error shade is the standard deviation.

all group actions is only feasible for smaller groups (rotation, reflection and scaling).

Further, Table 2 reports the performance obtained by a search algorithm applied on top of a hand-engineered domain-specific language (DSL). This approach searches all possible programs in the DSL that can map the input grids to the corresponding output grids successfully. We use the implementation of Wind (2020), which obtained the best results at the ARC Kaggle competition out of almost 1000 submissions [3]. This approach does not use any learnable component and the results are provided as a reference. We notice that LATFORMER significantly reduces the gap between neural networks and the current best approach for ARC, even outperforming the search algorithm for one category of tasks.

Though we restrict to only a subset of the tasks and there is definitely room for improvement even on these tasks, we reach considerably better performance than the baselines. Therefore, we believe our results advocate for the applicability of end-to-end differentiable models even on problems requiring sample-efficient abstract reasoning. To the extent of our knowledge, this is the first evidence of a neural network achieving this performance on ARC tasks.

5.3. Comparison with Neural Program Synthesis

Recently, Acquaviva et al. (2021) introduced the *Language-complete Abstraction and Reasoning Corpus* (LARC), which provides natural language descriptions of 88% of the ARC tasks, generated by human participants who where asked to communicate to other humans a set of precise instructions to solve a task.

Acquaviva et al. (2021) evaluated several models based on neural program synthesis on LARC. All models generate symbolic programs from a carefully designed domain-specific language (DSL) following a *generate-and-check* strategy. First a neural model generates a program from the grammar of the DSL (Ellis et al., 2020) and then the program is checked against the input-output pairs to ensure that it can generate all training examples.

We compare against the following baselines identified by Acquaviva et al. (2021). **LARC (IO)** is a model that has only access to input-output pairs, as our LATFORMER. **LARC (IO + NL)** has access to the natural language descriptions as well and uses a pre-trained T5 model (Raffel et al., 2020) to represent the text. **LARC (IO + NL pseudo)** uses *pseudo-annotation* to encourage the learning of compositional relationships between language and programs: during training, the model is given additional synthetic language-to-program pairs generated by annotating primitive examples in the DSL with linguistic comments. We refer the reader to Appendix B.3 for more details.

In order to compare to the work of Acquaviva et al. (2021), we evaluated their models on the set of LARC tasks that correspond to ARC tasks in our subset requiring geometric knowledge priors. Additionally, following Acquaviva et al. (2021) we allowed LATFORMER to access the textual descriptions by using a pre-trained T5 model to generate a representation of the text. This embedding is provided as input both to the *Lattice Mask Expert* and the *FFN* layers of LATFORMER (**LatFormer + NL**). Table 3 shows the results of our experiments on the LARC dataset. The program-synthesis methods require a training stage on a portion of the tasks. Therefore, the LATFORMER models where only evaluated on the same testing tasks of LARC, using the same train-test split of Acquaviva et al. (2021). Overall, our results show that LATFORMER performs better than program synthesis on the subset of tasks requiring geometric priors, with no need for a carefully designed DSL. This advantage comes to the expense of being restricted to tasks involving geometric priors, whereas program-synthesis approaches can be used on a wider set of tasks. We also observe that the natural language descriptions marginally helped our model on one category of tasks. Our findings corroborate with Acquaviva et al. (2021) in this remark.

[3] https://www.kaggle.com/competitions/abstraction-and-reasoning-challenge/

Table 2: Performance on ARC tasks that involve lattice symmetry priors.

	Translate	Rotate + Translate	Reflect + Translate	Scale + Translate
CNN	0.019	0.000	0.000	0.000
Attention (abs. pos.)	0.019	0.000	0.023	0.000
Attention (rel. pos.)	0.019	0.000	0.023	0.000
PixelCNN	0.019	0.000	0.000	0.000
Transformer	0.038	0.000	0.045	0.000
Differentiable Neural Computer	0.038	0.000	0.045	0.000
Transformer + data augmentation	-	0.200	0.184	0.091
LatFormer	0.365	**0.800**	0.591	0.545
Search over hand-crafted DSL	**0.673**	0.400	**0.614**	**0.727**

Table 3: Comparison of LATFORMER with neural program synthesis methods with access to both input-output pairs and natural language descriptions on LARC

	Translate	Rotate + Translate	Reflect + Translate	Scale + Translate
LARC (IO)	0.17	0.00	0.42	**1.00**
LARC (IO + NL)	0.17	0.00	0.42	**1.00**
LARC (IO + NL pseudo)	0.25	0.00	0.42	**1.00**
LatFormer	**0.33**	**1.00**	0.50	**1.00**
LatFormer + NL	**0.33**	**1.00**	**0.58**	**1.00**

6. Related Work

Our work was inspired by a previous investigation of self-attention layers which identified sufficient conditions such that they can perform convolution when equipped with relative positional encodings (Cordonnier et al., 2020; Andreoli, 2019). Rather than relying on relative encodings, we here show how soft-masking can be used to learn sample efficiently more general input transformations, such as rotation, reflection, and scaling.

To the extent of our knowledge, the group-action learning problem has not been explicitly and generally formulated in previous work. That being said, many previous works have focused on specific instances, such as learning to sort (Graves et al., 2014; Reed & De Freitas, 2015; Li et al., 2020) by selecting an element of the permutation group S_n, docking/folding by roto-translating objects according to an action in the special Euclidean group $\mathsf{SE}(3)$ (Sverrisson et al., 2022; Stärk et al., 2022; Jumper et al., 2021), and graph spectrum generation where the learned actions belong to the Stiefel manifold (Martinkus et al., 2022).

Our work is similar in spirit to recent efforts in neuro-symbolic visual reasoning (Johnson et al., 2017b;a; Goyal et al., 2017; Mao et al., 2019; Higgins et al., 2018) and in the area of enhancing machine-learning models with the ability to reason over structured data (Atzeni et al., 2021; Murugesan et al., 2021a; Atzeni & Atzori, 2018; Murugesan et al., 2021b). Many approaches based on attention mechanisms have been proposed in the past few years (Hudson & Manning, 2018; 2019). Our work differentiates from previous lines of research in that we aim to learn basic geometric reasoning in a sample-efficient way, rather than modeling relationships between high-level concepts.

Finally, some recent works came to our same conclusion on the advantages of using attention masks to incorporate prior knowledge in neural networks. As an example, Yan et al. (2020) focus on the task of learning subroutines (e.g., sorting algorithms) and use a CNN to generate an attention mask for a Transformer encoder. Similarly, Sartran et al. (2022) used precomputed attention masks to incorporate syntactic biases in language models.

7. Conclusion

Motivated by the long-term ambitious goal of infusing core knowledge priors in neural networks, this paper focused on how to help deep learning models to learn geometric transformations efficiently. Specifically, we proposed to incorporate lattice symmetry biases into attention mechanisms by modulating the attention weights using learned soft masks. We have shown that attention masks implementing the actions of the symmetry group of a hypercubic lattice exist, and we provided a way to represent these masks. This motivated us to introduce LATFORMER, a model that generates attention masks corresponding to lattice symmetry priors using a CNN. Our results on synthetic tasks show that our model can generalize better than the same attention modules without masking and Transformers. Moreover, the performance of our method on a subset of ARC provides the first evidence that deep learning can be used on this dataset, which is widely considered as an important open challenge for research on artificial intelligence.

References

Acquaviva, S., Pu, Y., Kryven, M., Wong, C., Ecanow, G. E., Nye, M. I., Sechopoulos, T., Tessler, M. H., and Tenenbaum, J. B. Communicating natural programs to humans and machines. *CoRR*, abs/2106.07824, 2021. URL https://arxiv.org/abs/2106.07824.

Andreoli, J.-M. Convolution, attention and structure embedding. *arXiv preprint arXiv:1905.01289*, 2019.

Atzeni, M. and Atzori, M. What is the cube root of 27? question answering over codeontology. In Vrandecic, D., Bontcheva, K., Suárez-Figueroa, M. C., Presutti, V., Celino, I., Sabou, M., Kaffee, L., and Simperl, E. (eds.), *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I*, volume 11136 of *Lecture Notes in Computer Science*, pp. 285–300. Springer, 2018. doi: 10.1007/978-3-030-00671-6_17. URL https://doi.org/10.1007/978-3-030-00671-6_17.

Atzeni, M., Bogojeska, J., and Loukas, A. SQALER: scaling question answering by decoupling multi-hop and logical reasoning. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 12587–12599, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/68bd22864919297c8c8a8c32378e89b4-Abstract.html.

Bahdanau, D., Murty, S., Noukhovitch, M., Nguyen, T. H., de Vries, H., and Courville, A. C. Systematic generalization: What is required and can it be learned? In *ICLR 2019*, 2019.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V. F., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gülçehre, Ç., Song, F., Ballard, A. J., Gilmer, J., Dahl, G. E., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., and Pascanu, R. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018.

Bengio, Y. The consciousness prior. *CoRR*, abs/1709.08568, 2017.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Chollet, F. On the measure of intelligence. *CoRR*, abs/1911.01547, 2019. URL http://arxiv.org/abs/1911.01547.

Cordonnier, J.-B., Loukas, A., and Jaggi, M. On the relationship between self-attention and convolutional layers. In *Eighth International Conference on Learning Representations-ICLR 2020*, number CONF, 2020.

Ellis, K., Wong, C., Nye, M. I., Sablé-Meyer, M., Cary, L., Morales, L., Hewitt, L. B., Solar-Lezama, A., and Tenenbaum, J. B. Dreamcoder: Growing generalizable, interpretable knowledge with wake-sleep bayesian program learning. *CoRR*, abs/2006.08381, 2020. URL https://arxiv.org/abs/2006.08381.

Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., and Parikh, D. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 6325–6334. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.670. URL https://doi.org/10.1109/CVPR.2017.670.

Graves, A., Wayne, G., and Danihelka, I. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwinska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J. P., Badia, A. P., Hermann, K. M., Zwols, Y., Ostrovski, G., Cain, A., King, H., Summerfield, C., Blunsom, P., Kavukcuoglu, K., and Hassabis, D. Hybrid computing using a neural network with dynamic external memory. *Nat.*, 538 (7626):471–476, 2016. doi: 10.1038/nature20101. URL https://doi.org/10.1038/nature20101.

Gul, M. S. K., Bätz, M., and Keinert, J. Pixel-wise confidences for stereo disparities using recurrent neural networks. In *30th British Machine Vision Conference 2019, BMVC 2019, Cardiff, UK, September 9-12, 2019*, pp. 23. BMVA Press, 2019. URL https://bmvc2019.org/wp-content/uploads/papers/0274-paper.pdf.

Higgins, I., Sonnerat, N., Matthey, L., Pal, A., Burgess, C. P., Bosnjak, M., Shanahan, M., Botvinick, M. M., Hassabis, D., and Lerchner, A. SCAN: learning hierarchical compositional visual concepts. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rkN2Il-RZ.

Hudson, D. A. and Manning, C. D. Compositional attention networks for machine reasoning. In *6th International Conference on Learning Representations,*

ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018. URL https://openreview.net/forum?id=S1Euwz-Rb.

Hudson, D. A. and Manning, C. D. Learning by abstraction: The neural state machine. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 5901–5914, 2019.

Jiang, W., Trulls, E., Hosang, J., Tagliasacchi, A., and Yi, K. M. COTR: correspondence transformer for matching across images. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pp. 6187–6197. IEEE, 2021. doi: 10.1109/ICCV48922.2021.00615. URL https://doi.org/10.1109/ICCV48922.2021.00615.

Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L., and Girshick, R. B. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 1988–1997. IEEE Computer Society, 2017a. doi: 10.1109/CVPR.2017.215. URL https://doi.org/10.1109/CVPR.2017.215.

Johnson, J., Hariharan, B., van der Maaten, L., Hoffman, J., Fei-Fei, L., Zitnick, C. L., and Girshick, R. B. Inferring and executing programs for visual reasoning. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 3008–3017. IEEE Computer Society, 2017b. doi: 10.1109/ICCV.2017.325. URL https://doi.org/10.1109/ICCV.2017.325.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Kolev, V., Georgiev, B., and Penkov, S. Neural abstract reasoner. *CoRR*, abs/2011.09860, 2020. URL https://arxiv.org/abs/2011.09860.

Lake, B. M. and Baroni, M. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *ICML*, 2018.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40:e253, 2017. doi: 10.1017/S0140525X16001837.

Li, Y., Gimeno, F., Kohli, P., and Vinyals, O. Strong generalization and efficiency in neural programs. *CoRR*, abs/2007.03629, 2020. URL https://arxiv.org/abs/2007.03629.

Lindblad, J. and Sladoje, N. Linear time distances between fuzzy sets with applications to pattern matching and classification. *IEEE Trans. Image Process.*, 23(1):126–136, 2014. doi: 10.1109/TIP.2013.2286904. URL https://doi.org/10.1109/TIP.2013.2286904.

Lowe, D. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pp. 1150–1157 vol.2, 1999. doi: 10.1109/ICCV.1999.790410.

Lu, J., Öfverstedt, J., Lindblad, J., and Sladoje, N. Is image-to-image translation the panacea for multimodal image registration? A comparative study. *CoRR*, abs/2103.16262, 2021. URL https://arxiv.org/abs/2103.16262.

Mao, J., Gan, C., Kohli, P., Tenenbaum, J. B., and Wu, J. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rJgMlhRctm.

Martinkus, K., Loukas, A., Perraudin, N., and Wattenhofer, R. Spectre: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *International Conference on Machine Learning*, ICML, 2022.

Murugesan, K., Atzeni, M., Kapanipathi, P., Shukla, P., Kumaravel, S., Tesauro, G., Talamadupula, K., Sachan, M., and Campbell, M. Text-based RL agents with commonsense knowledge: New challenges, environments and baselines. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 9018–9027. AAAI Press, 2021a. URL https://ojs.aaai.org/index.php/AAAI/article/view/17090.

Murugesan, K., Atzeni, M., Kapanipathi, P., Talamadupula, K., Sachan, M., and Campbell, M. Efficient text-based reinforcement learning by jointly leveraging state and commonsense graph representations. In Zong, C., Xia, F., Li, W., and Navigli, R. (eds.), *Proceedings of the*

59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 2: Short Papers), Virtual Event, August 1-6, 2021, pp. 719–725. Association for Computational Linguistics, 2021b. doi: 10.18653/v1/2021.acl-short.91. URL https://doi.org/10.18653/v1/2021.acl-short.91.

Pielawski, N., Wetzer, E., Öfverstedt, J., Lu, J., Wählby, C., Lindblad, J., and Sladoje, N. Comir: Contrastive multimodal image representation for registration. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Reed, S. and De Freitas, N. Neural programmer-interpreters. *arXiv preprint arXiv:1511.06279*, 2015.

Sarlin, P., DeTone, D., Malisiewicz, T., and Rabinovich, A. Superglue: Learning feature matching with graph neural networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 4937–4946. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.00499. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Sarlin_SuperGlue_Learning_Feature_Matching_With_Graph_Neural_Networks_CVPR_2020_paper.html.

Sartran, L., Barrett, S., Kuncoro, A., Stanojevic, M., Blunsom, P., and Dyer, C. Transformer grammars: Augmenting transformer language models with syntactic inductive biases at scale. *CoRR*, abs/2203.00633, 2022. doi: 10.48550/arXiv.2203.00633. URL https://doi.org/10.48550/arXiv.2203.00633.

Shaw, P., Uszkoreit, J., and Vaswani, A. Self-attention with relative position representations. In Walker, M. A., Ji, H., and Stent, A. (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pp. 464–468. Association for Computational Linguistics, 2018. doi: 10.18653/v1/n18-2074. URL https://doi.org/10.18653/v1/n18-2074.

Spelke, E. S. and Kinzler, K. D. Core knowledge. *Developmental Science*, 10(1):89–96, 2007. doi: https://doi.org/10.1111/j.1467-7687.2007.00569.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-7687.2007.00569.x.

Stärk, H., Ganea, O.-E., Pattanaik, L., Barzilay, R., and Jaakkola, T. Equibind: Geometric deep learning for drug binding structure prediction, 2022.

Sverrisson, F., Feydy, J., Southern, J., Bronstein, M. M., and Correia, B. Physics-informed deep neural network for rigid-body protein docking. In *ICLR2022 Machine Learning for Drug Discovery*, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017.

Wind, J. S. DSL solution to the ARC challenge, 2020. URL https://github.com/top-quarks/ARC-solution/blob/master/ARC-solution_documentation.pdf.

Yan, Y., Swersky, K., Koutra, D., Ranganathan, P., and Hashemi, M. Neural execution engines: Learning to execute subroutines. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

A. Additional Details on the Model

This section describes the LATFORMER architecture providing additional details that were not covered in Section 4.1. As mentioned in Section 4.1, it is possible to design convolutional neural networks that perform all considered transformations of the lattice. Figure 6 shows the architecture of the four expert models that generate *translation*, *rotation*, *reflection* and *scaling* masks.

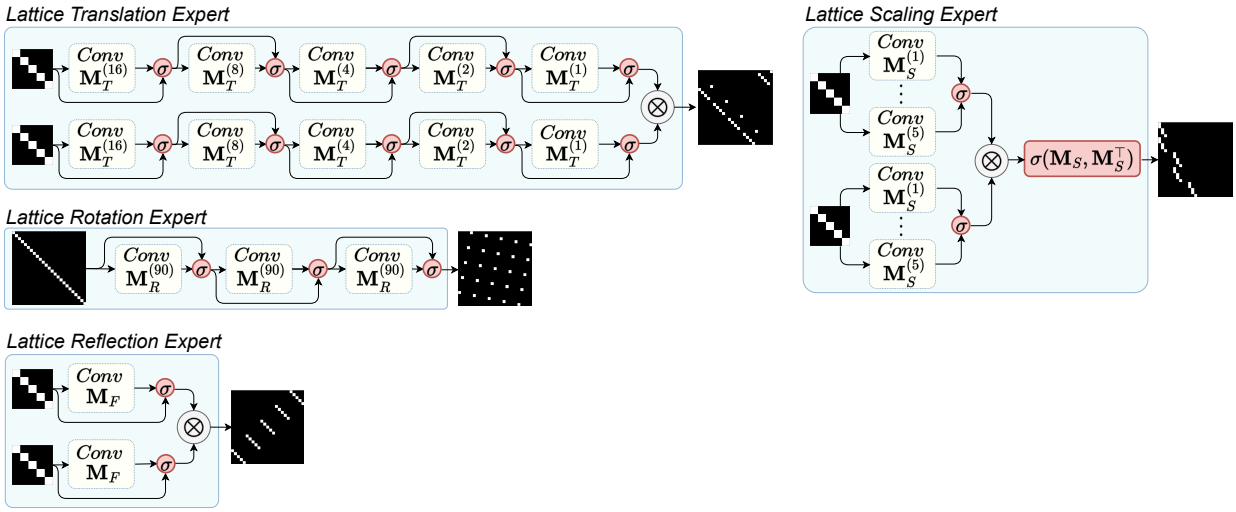


Figure 6: Model architecture of all the mask experts that we considered.

All models are CNNs applied to the identity matrix. In the figure, we use the following notation:

- $\boldsymbol{M}_T^{(\delta)}$ denotes an attention mask implementing a translation by δ along one dimension;
- $\boldsymbol{M}_R^{(90)}$ denotes an attention mask implementing a translation by $90°$;
- $\boldsymbol{M}_F$ denotes an attention mask implementing a reflection along one dimension;
- $\boldsymbol{M}_S^{(h)}$ denotes an attention mask implementing an upscaling by h along one dimension.

Using Corollary 3.3, we can derive the kernels of the convolutional layers shown in Figure 6. These kernels are frozen at training time, the model only learns the gating function, denoted as σ in the figure. Notice that all the models follow the same overall structure. However, for scaling, we also learn an additional gate, denoted as $\sigma(\boldsymbol{M}_S, \boldsymbol{M}_S^\top)$ in the Figure 6. This gate allows the model to transpose the mask and serves the purpose of implementing down-scaling operations (down-scaling is the transpose of up-scaling).

The composition of more actions can be obtained by combining different experts. This can be done either by chaining the experts or by matrix multiplication of the masks. In preliminary experiments, we did not notice any significant difference in performance between the two options and we rely on the latter in our implementation.

B. Additional Experiments and Details on the Experimental Setup

This section provides additional details on the experimental setup of all our experiments, including further information on the generation of the synthetic tasks and the data annotation process for ARC.

B.1. Experiments on Synthetic Data

We considered four categories of tasks, namely *translation*, *rotation*, *reflection* and *scaling*. Each task is defined in terms of input-output pairs, which are sampled from the set of all ARC grids and padded to the size of 30×30 cells. To each input

grid, a synthetic transformation is applied in order to obtain the corresponding output grid. For each task in each category, we generated 2048 training pairs and 100 test pairs.

For translation tasks, we have a total of 900 possible translations in a 30×30 grid. However, generating data and training models on 900 tasks is computationally expensive, so we randomly sampled 5 translations in the interval $[1, 29] \times [1, 29]$, obtaining a total of 100 translation tasks. Rotation tasks include all 4-fold rotations except the identity. Similarly, reflection tasks involve horizontal, vertical and diagonal reflections. Scaling tasks include all possible up/down scaling transformations of the input grid by factors of $[2, 5] \times [2, 5]$ for a total of 32 scaling tasks.

The models are evaluated based on the mean accuracy on each category. For each task we compute the accuracy on the test set based on how many of the predicted images match exactly the ground truth.

B.2. Experiments on ARC

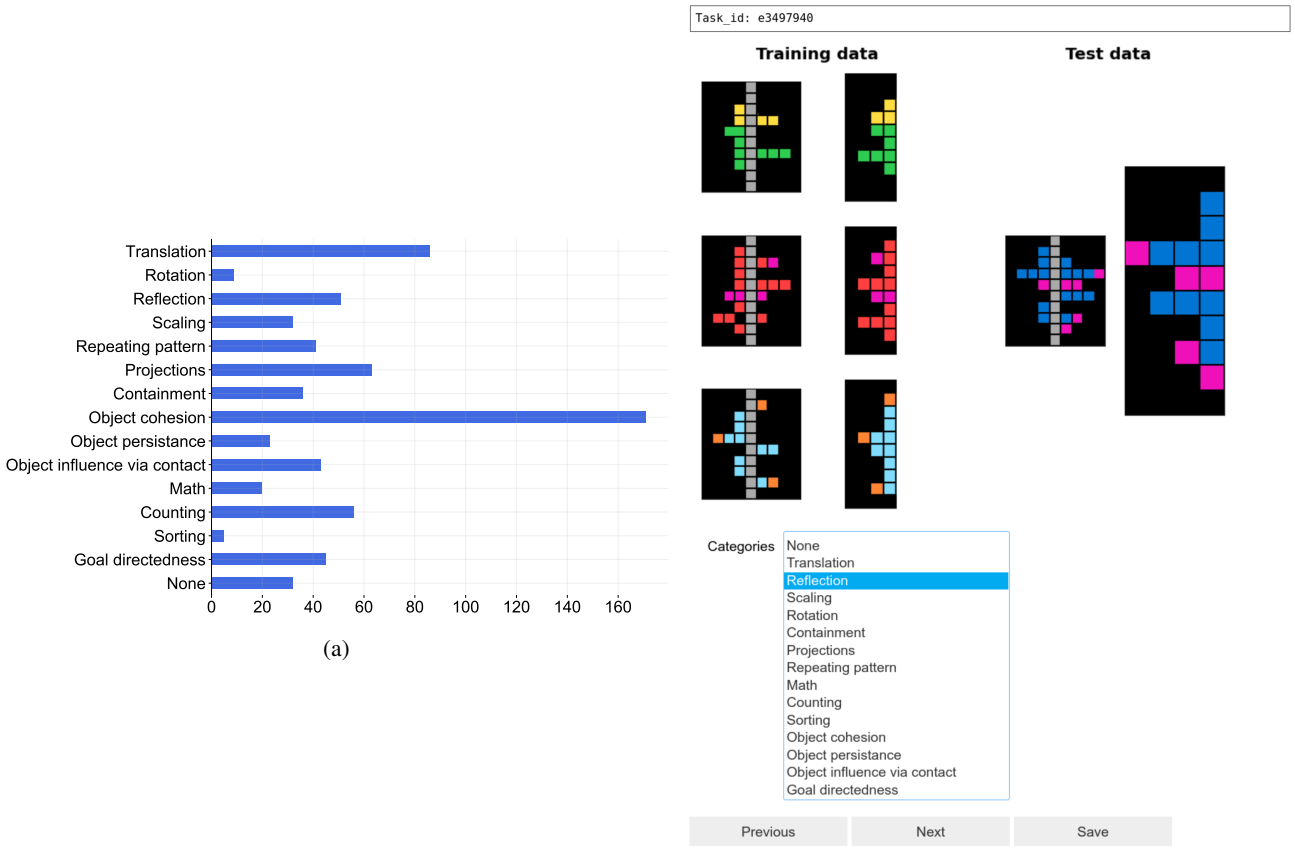


Figure 7: Distribution of the considered core knowledge priors across the ARC tasks (a) and user interface built to annotate the dataset (b).

In order to experiment with ARC, we first performed an annotation of the dataset to identify the underlying knowledge priors for each task. To this end, we built a user interface where the annotator could browse the tasks and label them by selecting any combination of the available knowledge priors. Figure 7b shows the user interface provided to the annotator, whereas Figure 7a shows the distribution of knowledge priors across the ARC tasks. Most tasks follow in more than one of the categories represented in Figure 7a.

ARC can be regarded as a meta-learning benchmark, as it provides a set of training tasks and a set of unseen tasks to evaluate the performance of the model learned on the meta-training data. It is important to emphasize that we do not target this use case, as we instead use the same setup as in the synthetic data and learn each task from scratch using only its training set.

Table 4: Results of the experiment on the robustness to noise

	Translation	**Rotation**	**Reflection**	**Scaling**
$w = 0.2$	0.91	1.00	1.00	1.00
$w = 0.4$	0.89	0.96	0.96	0.92
$w = 0.6$	0.62	0.69	0.68	0.65

Though simple and elegant, the supervised-learning formulation prevents our models from reusing knowledge that can be shared between different tasks. In order to mitigate this issue, we rely on a data-augmentation strategy. At training time, for each model and every iteration, we augment each grid 10 times by mapping each color to a different color (using the same mapping across training examples). The rationale behind this data-augmentation strategy is that *(1)* we assume that for tasks involving only geometric knowledge priors to be not affected by color mapping and *(2)* all models (including LATFORMER) need to learn a function from d-dimensional color representations to categorical variables, hence it is beneficial if all colors are represented in the training set.

All models are evaluated based on the ratio of solved tasks and a task is considered solved if the model can predict the correct output grid for *all* examples in the test set.

B.3. Experiments on LARC

All baselines relying on program synthesis for the experiment on LARC are taken from the work of Acquaviva et al. (2021). They share an underlying formulation based on the *generate-and-check* strategy. The program synthesizer generates a program *prog* given a natural program *natprog* (which can defined by either the input-output pairs alone or by input-output pairs and the corresponding natural language description) from the following distribution:

$$P_{synth}(prog \mid natprog) \propto P_{gen}(prog \mid natprog)\mathbb{1}[prog \vdash IO].$$

Above, P_{gen} is the generative distribution and $\mathbb{1}[prog \vdash IO]$ is the checker. The generative distribution proposes programs by first generating a tree bigram over the grammar of a DSL and then enumerating deterministically programs from a probabilistic context free grammar fitted to this bigram distribution in decreasing probability. For simplicity, Acquaviva et al. (2021) used an unconditioned generator $P_{synth}(prog)$ (i.e., a fitted prior) when language is absent, and language-conditioned models $P_{synth}(prog \mid NL)$ when a natural language description *NL* is given.

Once a program has been proposed, the checker validates the program *prog* by executing it on the interpreter, ensuring that $prog(x) = y$ for all input-output pairs $(x, y) \in IO$, where *IO* denotes the set of input-output pairs. The key strength of this approach lies in its generalizability, as programs that can be checked successfully on all training examples are likely to generalize.

B.4. Robustness to Noise

In order to assess the robustness of our method, we performed an additional experiment with synthetic tasks, where we introduced noise based on the following process. Input-output grids in our tasks contain categorical values from 1 to 10. In our experiments of Section 5, we represented each categorical value using an embedding layer, which essentially applies a linear transformation on a one-hot encoding of the categorical value. Given that we use one-hot vectors to represent categorical values, we apply noise directly to the one-hot vectors as follows:

$$\boldsymbol{x}_{noise} = (1 - w)\boldsymbol{W}^\top \mathbf{1}_{hot}(x) + wW^\top \mathbf{1}$$

where $w \in [0, 1]$ is the noise level, $\mathbf{1}_{hot}(x)$ is the one-hot encoding of a categorical value x, $\boldsymbol{W} \in \mathbb{R}^{n \times d}$ is a learnable matrix, n is the total number of categorical values (10 in our experiments) and $\boldsymbol{x}_{noise}$ is the noisy representation of the categorical value x. We evaluated the ability of LatFormer to denoise the input and apply the correct transformation at the same time on our synthetic tasks. Each task has a training set consisting of 32 examples and a test set consisting of 100 examples. The following table reports the accuracy for 3 different noise levels and shows the robustness of our method to noise.

Table 5: Results of the experiment on image registration. The rows represent different models trained to translate images from modality A to B ($A \to B$) or viceversa ($B \to A$).

	Aerial data			Cytological data		
	α-AMD	SIFT	LatFormer	α-AMD	SIFT	LatFormer
CycleGAN ($A \to B$)	5.3 ± 3.1	67.2 ± 16.8	$\mathbf{68.3 \pm 4.5}$	$\mathbf{74.2 \pm 3.8}$	30.2 ± 4.2	68.3 ± 2.2
CycleGAN ($B \to A$)	65.7 ± 6.7	84.0 ± 2.5	$\mathbf{86.1 \pm 3.1}$	21.3 ± 1.8	18.2 ± 3.5	$\mathbf{24.2 \pm 3.3}$
DRIT++ ($A \to B$)	35.3 ± 2.4	38.1 ± 8.1	$\mathbf{38.2 \pm 5.9}$	50.4 ± 12.1	24.2 ± 2.7	$\mathbf{62.7 \pm 10.2}$
DRIT++ ($B \to A$)	20.2 ± 2.1	38.3 ± 4.5	$\mathbf{43.2 \pm 4.1}$	$\mathbf{30.1 \pm 4.5}$	5.2 ± 3.1	15.6 ± 3.5
pixel2pixel ($A \to B$)	84.2 ± 4.0	$\mathbf{98.7 \pm 0.4}$	89.3 ± 2.2	53.2 ± 6.9	9.5 ± 1.0	$\mathbf{61.2 \pm 5.5}$
pixel2pixel ($B \to A$)	68.2 ± 7.5	87.5 ± 4.03	$\mathbf{89.7 \pm 3.3}$	0.2 ± 0.1	4.0 ± 1.0	$\mathbf{4.2 \pm 1.1}$
StarGAN ($A \to B$)	63.1 ± 7.8	7.4 ± 2.7	$\mathbf{72.2 \pm 6.3}$	$\mathbf{60.2 \pm 12.2}$	12.2 ± 2.0	59.5 ± 5.9
StarGAN ($B \to A$)	52.1 ± 4.0	7.9 ± 1.3	$\mathbf{53.3 \pm 4.0}$	$\mathbf{20.8 \pm 3.9}$	4.1 ± 0.9	13.4 ± 3.1
CoMIR	94.2 ± 5.7	$\mathbf{100.0 \pm 0.0}$	90.2 ± 3.3	76.2 ± 12.1	74.1 ± 6.3	$\mathbf{78.1 \pm 3.4}$

B.5. Additional Experiments on Image Registration

As an additional experiment, to assess the applicability of our LATFORMER on natural images, we performed experiments on multimodal *image registration*, namely the problem of spatially aligning images from different modalities. Image registration is a well-studied problem in computer vision and we do not aim to establish state-of-the-art performance. The main purpose of this experiment is giving a hint on the applicability of our method to natural images beyond ARC. We refer the reader to SuperGlue (Sarlin et al., 2020) and COTR (Jiang et al., 2021) to have a sense of approaches specifically designed for this task.

Popular approaches to multimodal image registrations work in two stages: first, they learn a model that converts one modality into the other (or to transfer both modalities in the same representation as proposed by Pielawski et al. (2020)), then they align the two images using traditional techniques. We follow the experimental setup of Lu et al. (2021) and experiment with two datasets, one containing aerial views of a urban neighborhood and one containing cytological images. The images we employ are views of the same scene, but they are taken with different modalities and they are translated with respect to one another. We use the code of the authors to generate data involving only translations. Lu et al. (2021) additionally consider small rotations, but these transformations are not actions in the symmetry group of a lattice, so we are not interested in resolving them.

We employ several state-of-the-art methods for modality translation and we compare our method to α-AMD (Lindblad & Sladoje, 2014) and SIFT (Lowe, 1999) based on the success rate metric defined by Lu et al. (2021). A registration is considered successful if the relative registration error (i.e., the residual distance between the reference patch and the transformed patch after registration normalized by the height and width of the patch) is below 2%. Table 5 reports our results on the image registration tasks and shows that our approach performs well on both datasets coupled with different methods for modality translation. We use the same models of Lu et al. (2021) for the modality translation task. Then, in order to solve the image registration task with LATFORMER, we divide each image into 30×30 patches and we run our model to predict the translation from one patch in an image to its counterpart in the corresponding image.

C. Limitations and Future Work

Although we believe our results are interesting and promising for learning group actions with neural networks, we would like to point out some limitations of our approach. First, our method is limited to actions on the symmetry group of the hypercubic lattice and it is not immediately extendable to other groups. For instance, though permutation matrices are still convolutions of the identity and they can be generated by a CNN, providing an architecture with predefined kernels that can compute any permutation matrix is not feasible. Second, the model is hard to fine-tune: we noticed that once the gates of the CNN have been trained, it is hard for the model to adapt to different actions.

We believe that both limitations can be addressed by still keeping the same overall idea of modulating attention weights using soft attention masks, possibly with a different parametrization of the masks. Future work will focus on this research direction and on extending our work to cover a wider set of the ARC tasks.

D. Deferred Proofs

We prove both Theorem 3.1 and 3.2 by induction on the dimensionality of the hypercubic lattice m.

D.1. Base Case for Theorems 1 and 2

First, it is useful to notice that whenever $\boldsymbol{M} \in \{0,1\}^{n\times n}$ has exactly a single 1 per row, in other words $\boldsymbol{M}\cdot\mathbf{1}_n = \mathbf{1}_n$, then, for any $\boldsymbol{X}\in\mathbb{R}^{n\times d}$

$$\begin{aligned}\text{MaskedAttention}(\boldsymbol{X};\boldsymbol{M}) &= \frac{\boldsymbol{A}}{\boldsymbol{A}\cdot\mathbf{1}_n\mathbf{1}_n^\top}\boldsymbol{X}\\ &= \frac{\text{softmax}\big(\frac{\boldsymbol{X}\boldsymbol{X}^\top}{\sqrt{d}}\big)\odot\boldsymbol{M}}{\text{softmax}\big(\frac{\boldsymbol{X}\boldsymbol{X}^\top}{\sqrt{d}}\big)\odot\boldsymbol{M}\cdot\mathbf{1}_n\mathbf{1}_n^\top}\boldsymbol{X}\\ &= \boldsymbol{M}\cdot\boldsymbol{X}.\end{aligned}$$

In order to prove the theorems, we need to show that, for any action $\mathsf{g}\in\mathsf{G}_1$, including translations, reflections and rotations, there exists a mask $\boldsymbol{M}_\mathsf{g}$ such that:

$$\text{MaskedAttention}(\boldsymbol{X};\boldsymbol{M}_\mathsf{g}) = \mathsf{g}\circ\boldsymbol{X}.$$

Let us consider different families of actions separately.

Translation. As mentioned in Section 3.2, in the 1-dimensional case, a translation by one element to the right for a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^\top$ is given by the circulant permutation matrix:

$$\boldsymbol{M} = \boldsymbol{M}_T^{(1)} = \begin{pmatrix}0 & 0 & 0 & \cdots & 1\\ 1 & 0 & 0 & \cdots & 0\\ 0 & 1 & 0 & \cdots & 0\\ \vdots & \ddots & \ddots & \ddots & \vdots\\ 0 & \cdots & 0 & 1 & 0\end{pmatrix}.$$

This holds because $\boldsymbol{M}_T^{(1)}\cdot\mathbf{1}_n = \mathbf{1}_n$, so:

$$\text{MaskedAttention}(\boldsymbol{x};\boldsymbol{M}_T^{(1)}) = \boldsymbol{M}_T^{(1)}\cdot\boldsymbol{x} = (x_n, x_1, x_2, \ldots, x_{n-1})^\top.$$

In general, a translation by δ elements is given by the circulant matrix $\boldsymbol{M}_T^{(\delta)} = (\boldsymbol{M}_T^{(1)})^\delta$. This follows directly from the properties of circulant permutation matrix $\boldsymbol{M}_T^{(\delta)}$. Therefore, we have a base case for Thereom 3.1, as we have shown that masks implementing translation operations exist in the 1-dimensional case and they are circulant permutation matrices.

For Theorem 3.2, simply notice that, by the time-shifting property of the Fourier transform:

$$\boldsymbol{M}_T^{(\delta)} = \mathcal{F}^{-1}\big(\mathcal{F}(\boldsymbol{I}_n)\,\exp(-\frac{2\pi j}{n}\,\boldsymbol{o}_T^{(\delta)}\,\boldsymbol{r}_n^\top)\big)\quad\text{where}\quad \boldsymbol{o}_T^{(\delta)} = \begin{pmatrix}-\delta\\ -\delta\\ \vdots\\ -\delta\end{pmatrix}.$$

Intuitively, the reader can notice that the equation above states that the circulant permutation matrix $\boldsymbol{M}_T^{(\delta)}$ can be obtained by shifting the rows of the identity by δ to the left. Thus, we provided a base case for Thereom 3.2 as well.

Reflection. In the 1-dimensional case, the reflection of a vector $\boldsymbol{x} = (x_0, x_1, \ldots, x_n)^\top$ is:

$$\text{MaskedAttention}(\boldsymbol{x};\boldsymbol{M}_F) = \boldsymbol{M}_F\cdot\boldsymbol{x} = (x_n, x_{n-1}, \ldots, x_2, x_1)^\top$$

with

$$\boldsymbol{M}_F = \begin{pmatrix} 0 & \cdots & 0 & 0 & 1 \\ 0 & \cdots & 0 & 1 & 0 \\ 0 & \cdots & 1 & 0 & 0 \\ \vdots & \cdots & \vdots & \vdots & \vdots \\ 1 & \cdots & 0 & 0 & 0 \end{pmatrix}.$$

The attention mask $\boldsymbol{M}_F$ can be obtained by shifting the rows of the identity matrix by:

$$\boldsymbol{o}_F = \begin{pmatrix} n-1 \\ n-3 \\ n-5 \\ \vdots \\ 1 \end{pmatrix}.$$

Therefore, by the time-shifting property of the Fourier transform we have:

$$\boldsymbol{M}_F = \mathcal{F}^{-1}\big(\mathcal{F}(\boldsymbol{I}_n)\, \exp(-\frac{2\pi j}{n}\, \boldsymbol{o}_F\, \boldsymbol{r}_n^\top)\big).$$

Rotation. Rotation (4-fold) is not defined in one dimension, so for a base case we need to consider the square lattice. Let $\boldsymbol{X} \in \mathbb{R}^{l_1 \cdot l_2}$ be a vectorized representaiton of a $n = l_1 \times l_2$ dimensional matrix. We need to define a vector $\boldsymbol{o}_R \in \mathbb{R}^n$ such that:

$$\boldsymbol{M}_R^{(90)} = \mathcal{F}^{-1}\big(\mathcal{F}(\boldsymbol{I}_n)\, \exp(-\frac{2\pi j}{n}\, \boldsymbol{o}_R\, \boldsymbol{r}_n^\top)\big)$$

is a rotation mask. Since rotation is a permutation of the identity, we know the vector exists. As $\boldsymbol{X}$ is vectorized, the $\boldsymbol{o}_R^{(90)}$ needs to take into account the size of the first dimension l_1. For example, in order to perform a rotation on a vectorized representation, we need to map the first element of $\boldsymbol{X}$ to the position $(l_1 - 1)$. The reader can check that the vector given by

$$(\boldsymbol{o}_R^{(90)})_k = k \cdot (l_1 - 1) - \lfloor (k-1)/l_1 \rfloor$$

satisfies the equation above.

Scaling. Although scaling is not a group action of the symmetry group of the lattice, we pointed out that it still can be defined within the same general formulation as the other transformations. We can take the 1-dimensional lattice as a base case and consider a vector $\boldsymbol{x} = (x_0, x_1, \ldots, x_n)^\top$. Let $h \in \mathbb{N}$ be a parameter specifying the filter size of the scaling operation. As an example, for $h = 2$, we have:

$$\text{MaskedAttention}(\boldsymbol{x}; \boldsymbol{M}_S^{(h)}) = \boldsymbol{M}_S^{(h)} \cdot \boldsymbol{x} = (x_1, x_1, x_2, x_2, \ldots, x_{\lfloor n/2 \rfloor})^\top,$$

where:

$$\boldsymbol{M}_S^{(h)} = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix}.$$

This kind of matrix can also be obtained by shifting the rows of the identity as follows:

$$\boldsymbol{M}_S^{(h)} = \mathcal{F}^{-1}\big(\mathcal{F}(\boldsymbol{I}_n)\, \exp(-\frac{2\pi j}{n}\, \boldsymbol{o}_S^{(h)}\, \boldsymbol{r}_n^\top)\big),$$

where $(\boldsymbol{o}_S^{(h)})_k = (k - 1 \bmod h) + (h-1) \cdot \lfloor (k-1)/h \rfloor$.

D.2. Inductive Step for Theorems 1 and 2

Suppose that $\boldsymbol{M}_{\mathsf{g}_1} \in \{0,1\}^{n_1 \times n_1}$ and $\boldsymbol{M}_{\mathsf{g}_2} \in \{0,1\}^{n_2 \times n_2}$ are attention masks implementing actions $\mathsf{g}_1 \in \mathsf{G}_{m_1}$ and $\mathsf{g}_2 \in \mathsf{G}_{m_2}$ on some tensors $\mathbf{X}_1 \in \mathbb{R}^{l_1 \times \cdots \times l_{m_1}}$ and $\mathbf{X}_2 \in \mathbb{R}^{l'_1 \times \cdots \times l'_{m_2}}$, with $n_1 = l_1 \cdot \ldots \cdot l_{m_1}$ and $n_2 = l'_1 \cdot \ldots \cdot l'_{m_2}$. Consider a tensor $\mathbf{X} \in \mathbb{R}^{l_1 \times \cdots \times l_{m_1} \times l'_1 \times \cdots \times l'_{m_2}}$ and its vectorization $\boldsymbol{X} \in \mathbb{R}^n$ with $n = n_1 n_2$.

We have:

$$
\begin{aligned}
&\mathrm{MaskedAttention}(\boldsymbol{X}; \boldsymbol{M}_{\mathsf{g}_1} \otimes \boldsymbol{M}_{\mathsf{g}_2}) = \\
&\quad = (\boldsymbol{M}_{\mathsf{g}_1} \otimes \boldsymbol{M}_{\mathsf{g}_2})\, \boldsymbol{X} \\
&\quad = (\boldsymbol{M}_{\mathsf{g}_1} \otimes \boldsymbol{I}_{n_2})(\boldsymbol{I}_{n_1} \otimes \boldsymbol{M}_{\mathsf{g}_2}) \boldsymbol{X} \\
&\quad = \mathrm{MaskedAttention}(\mathrm{MaskedAttention}(\boldsymbol{X}; \boldsymbol{I}_{n_1} \otimes \boldsymbol{M}_{\mathsf{g}_2}); \boldsymbol{M}_{\mathsf{g}_1} \otimes \boldsymbol{I}_{n_2}).
\end{aligned}
$$

Now notice that:

$$
\begin{aligned}
\mathrm{MaskedAttention}(\boldsymbol{X}; \boldsymbol{I}_{n_1} \otimes \boldsymbol{M}_{\mathsf{g}_2}) &= (\boldsymbol{I}_{n_1} \otimes \boldsymbol{M}_{\mathsf{g}_2})\, \boldsymbol{X} \\
&= \mathrm{vec}(\boldsymbol{M}_{\mathsf{g}_2}\, \mathrm{vec}^{-1}(\boldsymbol{X})\, \boldsymbol{I}_{n_1}) \\
&= \mathrm{vec}(\boldsymbol{M}_{\mathsf{g}_2}\, \mathrm{vec}^{-1}(\boldsymbol{X})),
\end{aligned}
$$

and similarly

$$
\begin{aligned}
\mathrm{MaskedAttention}(\boldsymbol{X}; \boldsymbol{M}_{\mathsf{g}_1} \otimes \boldsymbol{I}_{n_2}) &= (\boldsymbol{M}_{\mathsf{g}_1} \otimes \boldsymbol{I}_{n_2})\, \boldsymbol{X} \\
&= \mathrm{vec}(\boldsymbol{I}_{n_2}\, \mathrm{vec}^{-1}(\boldsymbol{X})\, \boldsymbol{M}_{\mathsf{g}_1}^\top) \\
&= \mathrm{vec}((\boldsymbol{M}_{\mathsf{g}_1}\, \mathrm{vec}^{-1}(\boldsymbol{X})^\top)^\top).
\end{aligned}
$$

Therefore, we conclude that performing masked attention with the mask $\boldsymbol{M}_{\mathsf{g}_1} \otimes \boldsymbol{M}_{\mathsf{g}_2}$ on $\mathbf{X}$ is equivalent to applying g_1 on the first m_1 dimensions and g_2 on the last m_2 dimensions of $\mathbf{X}$. This provides a way for building attention masks for higher-dimensional lattices using the primitive masks defined in Section D.1, proving both Theorem 3.1 and 3.2.

D.3. Proof of Corollary 1

The proof of Corollary 1 follows immediately from Theorem 3.2 and from the property of the Fourier transform according to which multiplying in the Fourier domain implements a convolution in the original domain.