Solving Visual Analogies Using Neural Algorithmic Reasoning (Student Abstract)

Atharv Sonwane[1], Gautam Shroff[2], Lovekesh Vig[2],
Ashwin Srinivasan[1], Tirtharaj Dash[1]

[1] APPCAIR, BITS Pilani, K K Birla Goa Campus
[2] TCS Research, New Delhi
f20181021@goa.bits-pilani.ac.in, gautam.shroff@tcs.com, lovekesh.vig@tcs.com,
ashwin@goa.bits-pilani.ac.in, tirtharaj@goa.bits-pilani.ac.in

Abstract

We consider a class of visual analogical reasoning problems that involve discovering the sequence of transformations by which pairs of input/output images are related, so as to analogously transform future inputs. This program synthesis task can be easily solved via symbolic search. Using a variation of the 'neural analogical reasoning' approach, we instead search for a sequence of elementary neural network transformations that manipulate distributed representations derived from a symbolic space, to which input images are directly encoded. We evaluate the extent to which our 'neural reasoning' approach generalises for images with unseen shapes and positions.

Introduction

We consider a class of simple visual reasoning tasks specified by pairs of (input, output) images, all of which are related by the same unknown transformation procedure. Given a new image, the task is to generate the correct output in an analogous manner to the examples provided. We model this task as a program synthesis problem, where the mapping between input and output is represented by a composition of elementary transformations. Building on previous program synthesis approaches which construct complex programs from simpler primitives, we employ a variation on the neural algorithmic reasoning (Veličković and Blundell 2021) framework to replace elementary symbolic transformations with equivalent neural networks, and examine if this leads to the generalisation which is the basis of analogical reasoning.

We decouple representation learning from transform learning, first learning a latent representation that captures key information from the input followed by learning transforms inside this representation space for which input-output examples have been given. Fig 1 provides an overview of our approach wherein we search over possible combinations of primitives until one is found that satisfies all the example pairs. This solution is thus a neural algorithm, i.e., a composition of elementary neural networks, rather than a symbolic program, corresponding to the *unknown* analogical relation, which we can then apply to any query image to obtain an analogous result (output image).

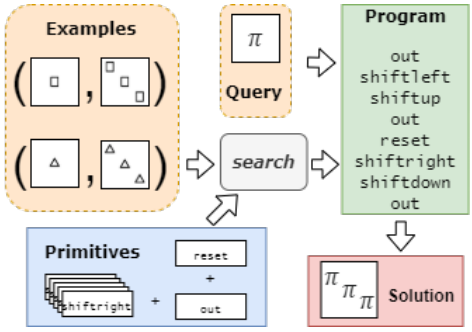

Figure 1: Overview

Methodology

The images considered consist of simple shapes (from a 20 member set of alphabets and polygons) placed on a 3×3 grid while positional shifts and shape conversions make up the elementary transforms with which to compose our solutions.

Representation Learning Let a latent space be characterised by an encoder E that maps natural inputs (e.g. images) to their latent representations (real-valued vectors). Since we are interested in spaces that are rich enough to represent a wide variety of shapes and sparse enough to represent concepts distinctly, we first construct a space based on symbolic descriptions of the input images containing information on the shapes and their positions in the 2D grid. For this, we train an autoencoder (Encoder: E_s, Decoder: D_s on the multi-hot (Boolean) vector representations of symbolic descriptions by minimising the negative log-likelihood loss between the inputs and the reconstructed vectors.

To be able to handle image inputs (denoted by xs), we train a CNN based encoder E_x to fit to the outputs of E_s (with frozen weights) by minimising the mean-squared-error over the encodings of images $E_x(x^{(i)})$ and encodings of the corresponding symbolic vectors $E_s(s^{(i)})$: MSE $\left(E_x(x^{(i)}), E_s(s^{(i)})\right)$.

To be able to handle shapes at test time that were not seen during training, we also include an additional shape-label called *unseen*. We then train E_x to map some set of shapes to the latent vectors generated by E_s corresponding to the *unseen* labels.

Transform Training We use single hidden layer MLPs as the neural transform networks $\mathcal{T}_i$s that manipulate the latent representations of the input images where a transform refers to some spatial transformations on the original space (images). For our experiments, we consider positional shifts in the ordinal directions (e.g. `shift-right` moves each shape in the image one grid-position to the right) and shape conversions (e.g. `to-square` converts each shape in the image to a square). We construct a separate transform network $\mathcal{T}_i$ for each such spatial transformation, and learn its weights by minimising the negative log-likelihood between $D_s(\mathcal{T}(E_s(s^{(i)}))$ and the multi-hot vector corresponding to the symbolic description of the transformed image, while keeping the encoder and decoder weights frozen.

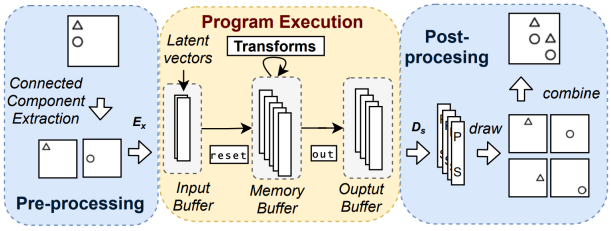

Figure 2: Pipeline

Program Evaluation We define primitives as transforms or special operations (`reset` and `out`); programs are sequences of such primitives. As shown in Fig. 2, the input image is preprocessed to extract individual shapes by identifying the connected components. Each isolated image is converted into a latent vector using the encoder E_x and stored in the input buffer. Algorithm 1 is applied and the resultant latent vectors in the output buffer are converted to their symbolic descriptions using D_s. The final image is constructed by drawing the corresponding shapes and combining them into a single image. (In the case of shapes marked as *unseen* in the symbolic description, we keep track of the original shape from the input and use it to generate the output.)

Algorithm 1: Program Execution

Input: Set of latent vectors z_ks for input, program P
Output: Set of latent vectors for output

Let input = $\{z_k\}$, memory = $\emptyset$, output = $\emptyset$
for $i = 1 :$ length(P) **do**
  if $P[i]$ = `out` **then**
    output $\leftarrow$ output $\cup$ memory
  else if $P[i]$ = `reset` **then**
    memory $\leftarrow$ input
  else
    memory $\leftarrow \{P[i](z_k), \forall z_k \in \text{memory}\}$
return output

Searching for Solution Given an (input, output) pair of images, we find the program that maps each input shape to the corresponding shape in the output by performing an exhaustive search over sequences of primitives. Various pruning measures are built into the search to make it tractable.

Results and Discussion

We evaluate our proposed system using ratio of (input, output) pairs for which a valid program is found. These pairs are generated from a dataset of 16743 programs (1000 programs for each length up to 20 with redundant programs removed). Fig. 3 suggests that our system is able to find the correct solutions for the complete dataset as well as all programs containing direct compositions of transforms up to length 5, demonstrating that conceptual integrity is maintained throughout successive transform applications. This is not the case when using a latent space based on an autoencoder trained on images as opposed to multi-hot vectors, suggesting a well-structured embedding space is particularly important in this setting.

We evaluate how well transforms $\mathcal{T}_i$s and image encoder E_x can generalise by withholding shapes during training and including them during evaluation. Note that when evaluating *transforms* with varying number of seen shapes, the *encoder* and *decoder* have seen all 20 shapes during training. When evaluating the encoder, the *unseen* label is used in addition to the labels corresponding to images seen during transform training; for this experiment, of the 20 shapes available, 4 have labels included in the symbolic descriptions while the rest are considered to have the label *unseen*. From Fig. 3, we see that performance improves with more shapes seen during training (as expected). However, the system is able to generalise since it finds a valid programs a significant portion of the time even when observing very few shapes during training. We observe similar results when evaluating generalisation to unseen grid-positions occupied by the shapes.

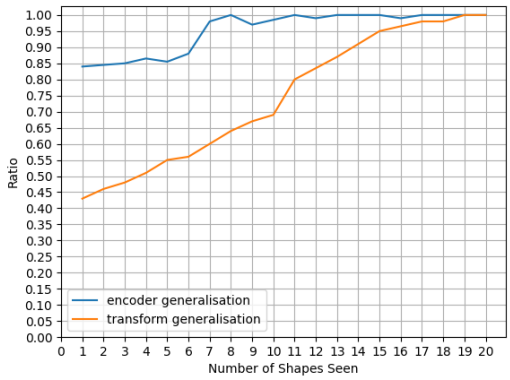

Figure 3: Generalisation Experiments

While our approach is limited by the use of classical algorithms for shape extraction as well as the mechanism for handling unseen shapes within a fixed symbolic code, we demonstrate within-domain generalisation capabilities (shapes seen during training the encoder but not transforms). Our approach, based on manipulating latent space representations, is, we submit, a step towards fully neural analogical reasoning with out-of-domain generalisation.

References

Veličković, P.; and Blundell, C. 2021. Neural algorithmic reasoning. *Patterns*, 2(7): 100273.