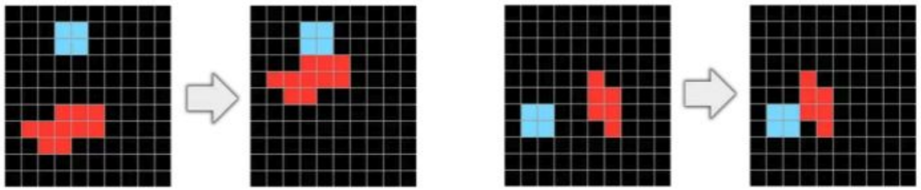
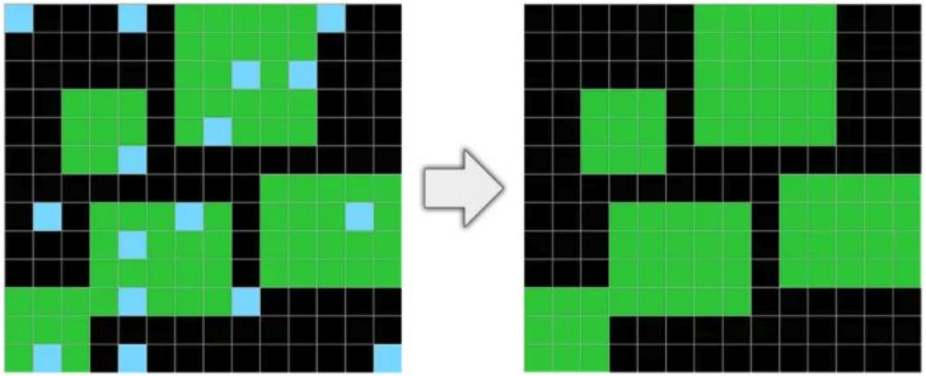
EE763
Abstract Reasoning Challenge

Mithilesh Vaidya
17D070011

Examples [5]



A training example in the ARC dataset: The red object must be made adjacent to the blue box.



ARC problem: the test taker must denoise the image, removing the blue dots (or any other color they might have) and keep the main objects intact.

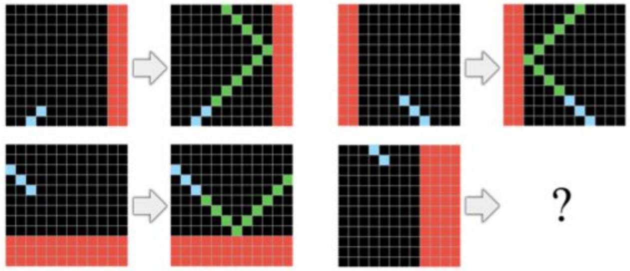
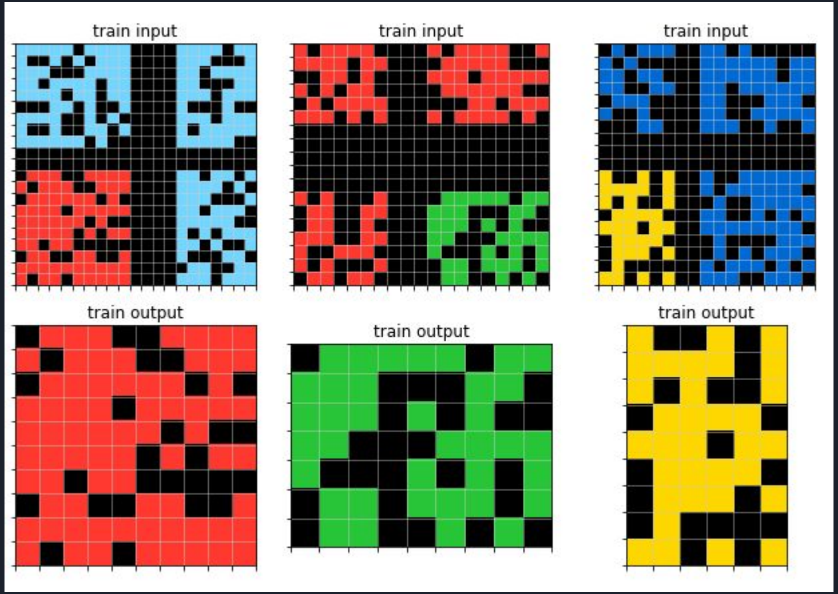
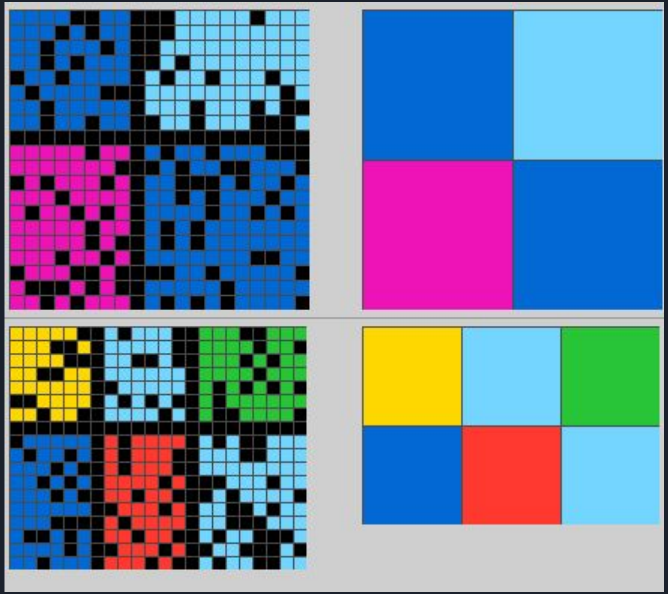


Figure 8: A task where the implicit goal is to extrapolate a diagonal line that “rebounds” upon contact with a red obstacle.

A few more

Size of input and output grid need not be the same! Have not dealt with such cases in this work





Introduction

- Most of the progress in DL is on very specific tasks e.g. object recognition, ASR, etc.
- Humans are amazing at inferring patterns with only a couple of examples
- Author hypothesis [1] that any computational model which can solve this dataset is closer to one aspect of general intelligence
- 400 train tasks with average 2.3 examples per task
- 600 test: 400 known and 200 unknown!
- Evaluation binary: entire grid must match
  We have 3 tries and feedback is also purely binary
- Difficult for DL due to limited size of dataset

Priors

Model should be powerful enough to capture:

- Objectness:
  - Cohesion: parse grids into objects based on continuity of colour and space
  - Persistence: persist despite e.g. in presence of noise
  - Contact: translate till one comes in contact with the other
- Goal-directedness: start and end states of a process
- Numbers and counting: count/sort objects by size
- Geometry and Topology:
  - Lines, rectangles, other shapes
  - Symmetries, rotation, scaling, translation

Could these be prerequisites for abstract reasoning and intelligence!?

Proposed models

- Domain-specific language (DSL)
  - Define basic set of operations e.g. split across colours, sort, invert colour, reorder, etc.
  - Chain many of these using genetic algorithms
- Deep Learning
  - Simple conv + activation over the grid
  - Use data augmentation to generate more samples
- Cellular Automata
  - Different CAs for different tasks; rules are hard-coded
  - Goal would be to automatically learn CA rules
- Best Kaggle solution
  - Similar to DSL
  - Apply anywhere between 4 to 142 unary transforms on the image
  - Solve for output size separately
  - Use a DAG for traversing the 142-depth tree

Cellular Automata (CA) [2]

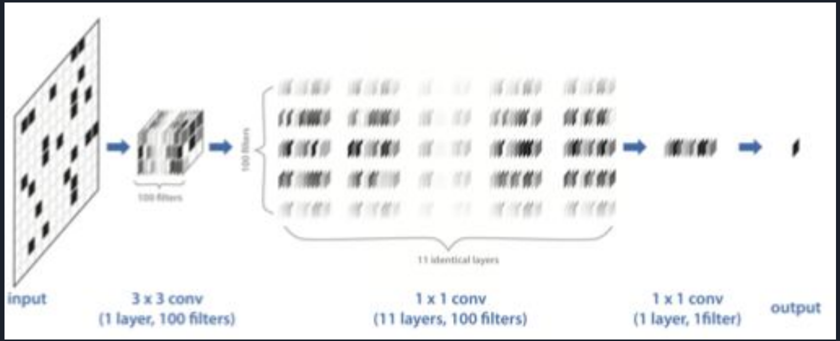
A system of cells which:

- Live on a grid (1D/2D)
- Each cell has a state and a neighborhood
- Update rule is global

Properties:

- Simple rules giving rise to complex patterns
- Shown to be Turing Universal

CAs as CNNs [3]



CA and CNN filters have:

- Locality of dynamics
- Simultaneous temporal update of cells

Paper shows that **any** CA rule can be implemented using CNNs

If each cell has M states and D neighbours, a total of M^D possible neighbourhoods

Gives us M^(M^D) possible *update rules*

Using only 3x3 filters, followed by many 1x1, paper argues any update rule can be learned from this large space

In our case, we only have start and end states and not the intermediate history -> we run the CA for a random number of steps

My experiments

8 Conv2D filters of kernel 5x5 + ReLU + 1 linear layer from 8 filters to number of colours

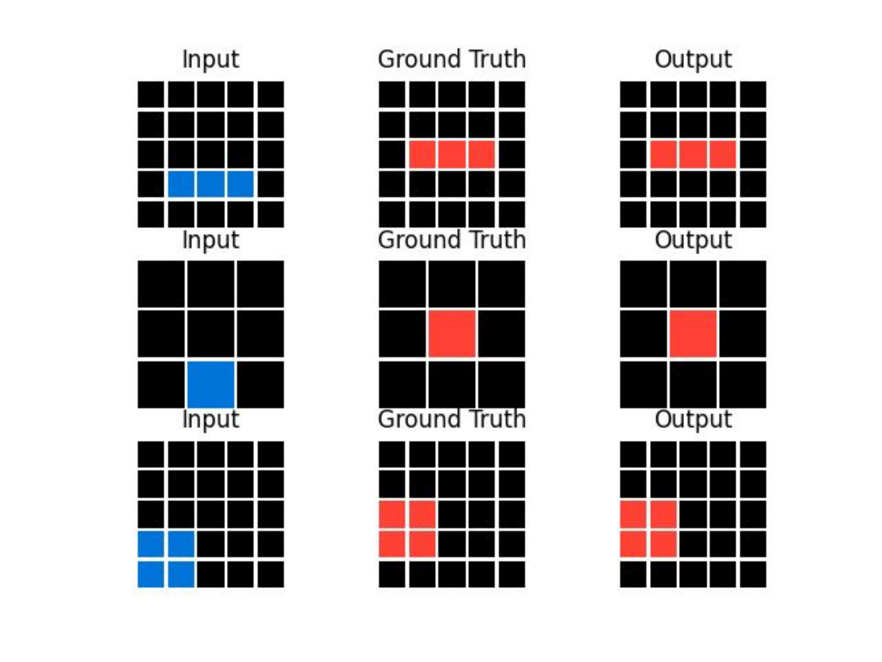
e.g. if there are 4 colours:
8 conv filters of shape (4, 5, 5): 800 (filter weights) + 8 (bias) = 808 params

1 linear layer of dimensions (8+4, 4) -> 48 + 4 = 52 params

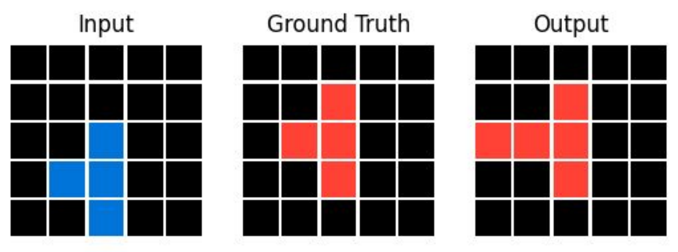
Total: 860 parameters only!

- At each step: predict delta (we also think of changes?)
- Added data augmentation: flip across rows, columns and both to get 3x dataset (exploiting known symmetry of rectangular grids. But is it valid in the long run?)
- Colours are one-hot encoded by sorting according to its count (we also don't care about the actual colour; more about its relative count)
- Training for 400 epochs and [5, 10] batch steps in recurrent fashion

Results

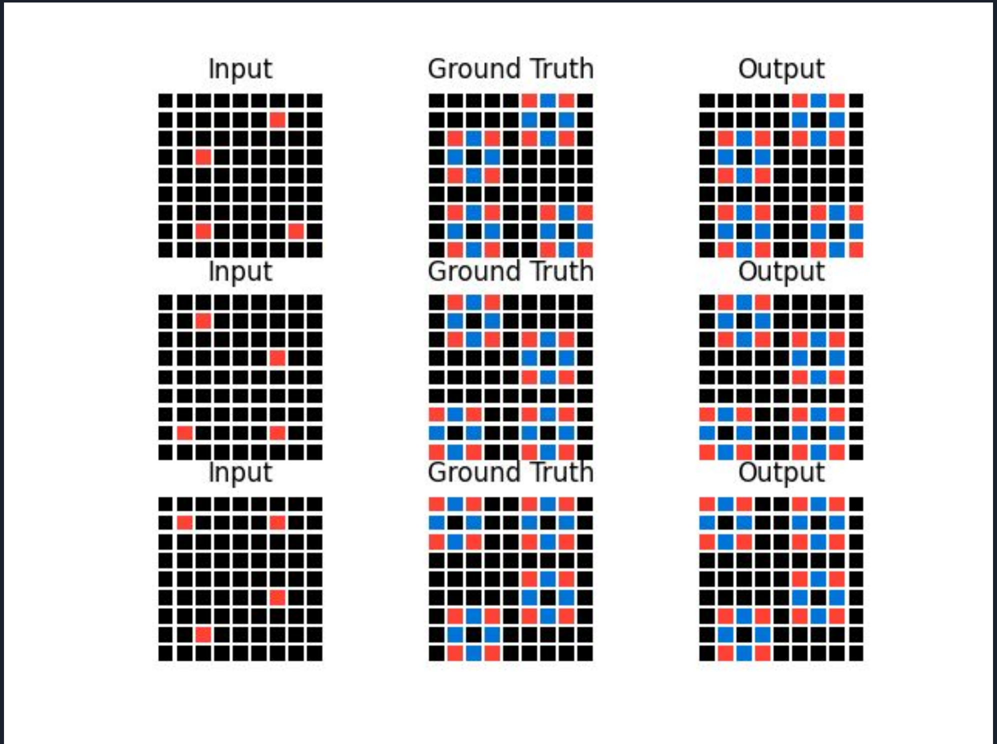


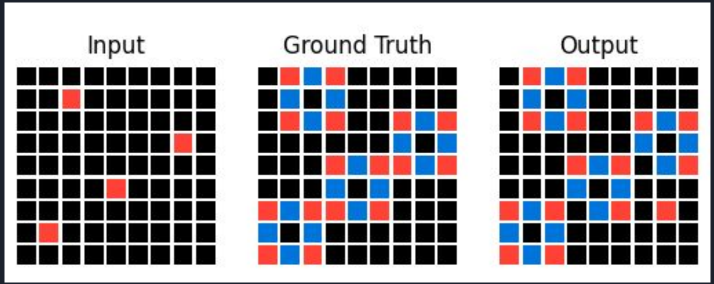
No data augmentation along vertical since we want to move it upwards ONLY



1 additional red square

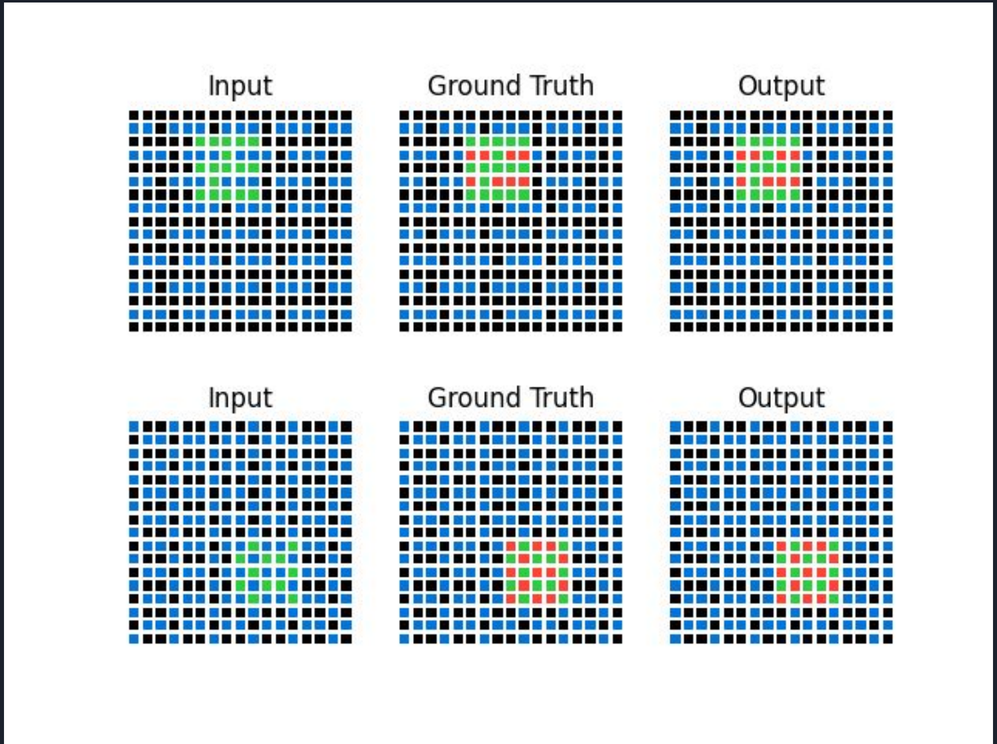
Results

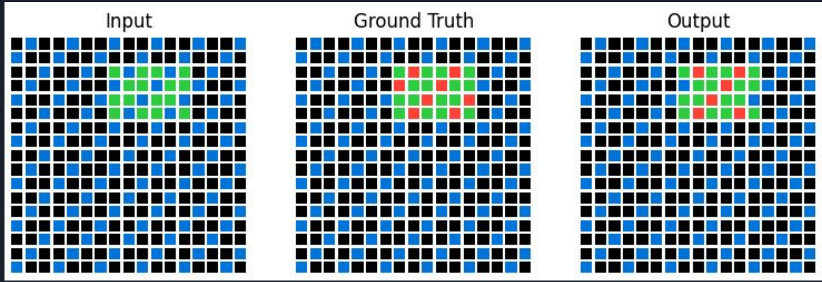




1 additional red square, 2 missing

Results

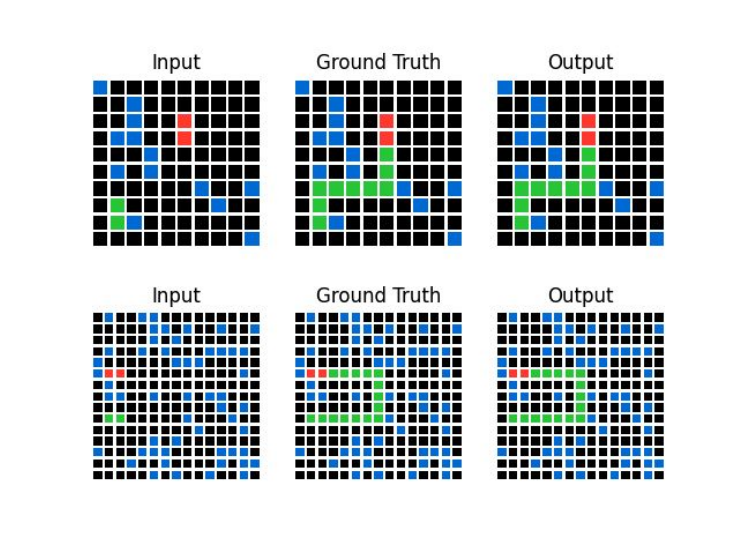


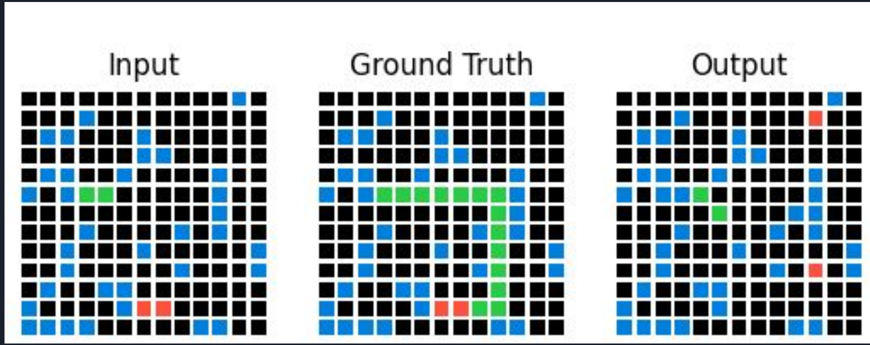


2 blue squares not replaced with red

Where it fails

- Added 2 more Conv2D filters
- Increased number of recurrent steps to [15, 20]





Completely random output

Reasons

- Previous task is of *sequential nature*
- In all other tasks, local neighbourhood knowledge was sufficient
- CA has no information about global state
- Information can flow from one cell to the other but in an inefficient, slow manner
- Need some global state which is provided to each cell

Comment on Kaggle:
CA can be a language for solving (describing solution), but not for actual reasoning. Reasoning requires analyzing and comparing input and output data to understand what needs to be done.

Once we know what to do, CAs can do it for us
But can they directly learn general rules? Or do we need to implicitly supply some *state*?

Possible extension

I was thinking of making *attractors* i.e. cells which give direction to other cells

Inspired by vector fields of charged particles in space:
can important cells act as charges?
The generated vector field can be provided as input to the CA

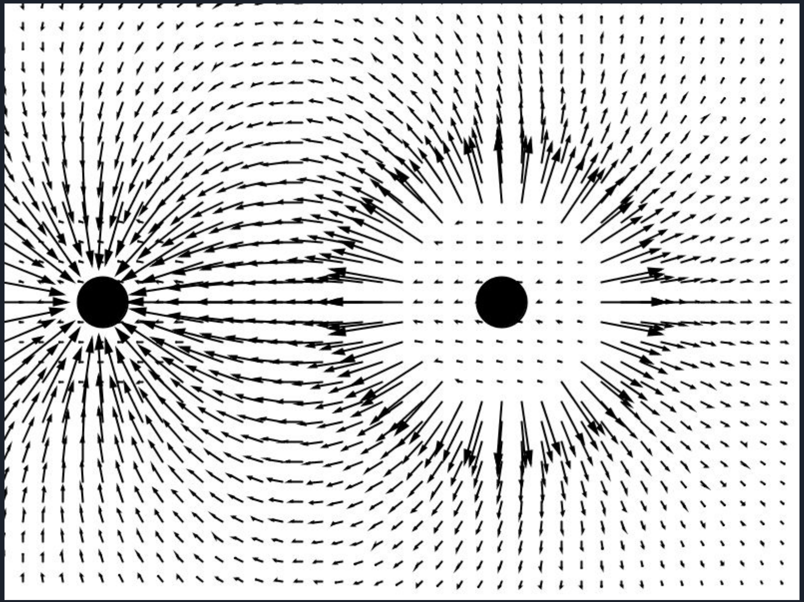


Figure from [4]

References

[1] On the Measure of Intelligence

[2] The Nature of Code

[3] [1809.02942] Cellular automata as convolutional neural networks

[4] Two point charges in Processing

[5] fchollet/ARC: The Abstraction and Reasoning Corpus

Grade

10

- One of the active participants in class
- Decent progress on new task after reviewing biology papers for previous SMART goals