Dataset description

The dataset is based on the Game of Life by J. Conway. It was created by running 52 games of life. Each game was run on a 2D grid 30 cells wide and 30 cells high. Each game was run for 300 generations (or steps, as we call them in the dataset).

In order to run a game, first a 30x30 grid was initialized, where each cell has a random probability to be "alive" (equal to 1) or "dead" (equal to 0), random probability ranges from 20% to 50%. Afterward, the rules of the Game of Life were iteratively applied to each state[1]:

- Any live cell with two or three live neighbors survives.
- Any dead cell with three live neighbors becomes a live cell.
- All other live cells die in the next generation. Similarly, all other dead cells stay dead.

The rows of the dataset correspond to a particular generation (step) of a particular game, the columns correspond to the status of each cell of the grid, e.g. the table below shows that the cell in loc [0, 13] in the 1st game on the 10th step was alive.

game_n	step	0_13
1	10	1

Apart from the cells states, the dataset also contains some metadata about the games/states:

- stable (bool): true if the current state of the game is exactly the same as the previous state (all dead and alive cells are in the same locations as the previous generation);
- period (bool): true if the current state of the game is the same as one of the previous states - if yes, that means that the game has entered a loop;
- live_cells_count (int): how many cells are alive in the current game and generation;
- alive (bool): true if there is at least one living cell in the current game and generation;
- alive_stable (bool): true if there is at least one living cell and the current generation is exactly the same as the previous one (true if "stable" == 1 and "alive" == 1);
- alive_period (bool): true if there is at least one living cell and the current generation is the same as one of the previous ones (true if "period" == 1 and "alive" == 1);
- game_alive (bool): true if a game ends up as "alive" – at the last step there is at least one alive cell;
- game_stable (bool): true if a game ends up in a stable state;

[1] *Conway's Game of Life*. Wikipedia. https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

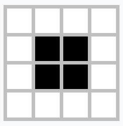
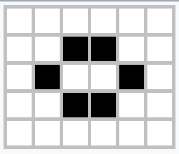
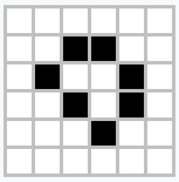
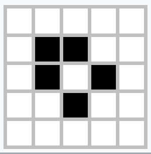
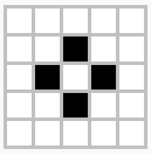
- game_period (bool): true if a game ends up as periodic;
- game_initial_cell_count (int): the number of alive cells at the initialization of the game.

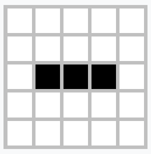
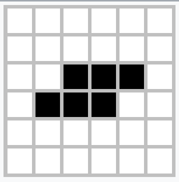
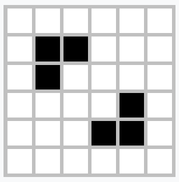
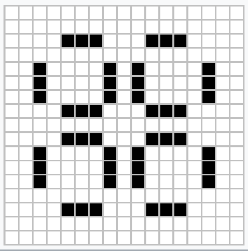
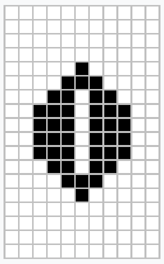
Together with the metadata, the shape of the dataset is (15600, 913):

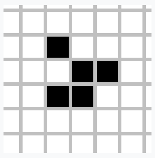
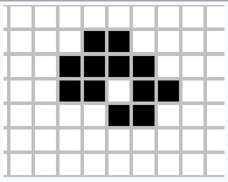
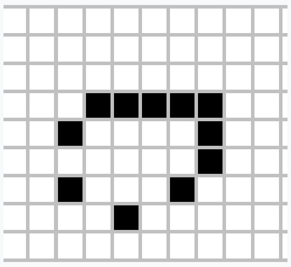
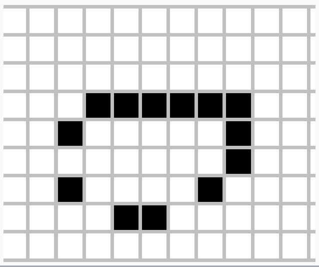
- 52 games x 300 steps = 15600;
- 30 height x 30 width + 13 metadata.

Patterns:

After a sufficient number of generations, if the grid still contains alive cells, we might see some of the following patterns appearing on the grid:

Still lifes	
Block	
Bee-hive	
Loaf	
Boat	
Tub	

Oscillators	
Blinker (period 2)	
Toad (period 2)	
Beacon (period 2)	
Pulsar (period 3)	
Penta-decathlon (period 15)	

Spaceships	
Glider	
Light-weight spaceship (LWSS)	
Middle-weight spaceship (MWSS)	
Heavy-weight spaceship (HWSS)	

Possible States:

Each cell in a grid can either be dead or alive, therefore there are only two possible outcomes for one cell.
For a $n \times n$ grid, the possible state space is $2^{n \times n} = 2^{n^2}$, e.g. for 2×2 it is $2^{2^2} = 2^4 = 16$ and for 30×30 it is $2^{30^2} = 2^{900} = 8.45\mathrm{e}270$.

We can also compare that with other state spaces, like the Rubik's Cube where the solution space is $4.3\mathrm{e}19$ (as stated by Hinterreiter et al. in the **Projection Path Explorer**)[2]:

- For Go with board size 19×19, it is $1\mathrm{e}172$
- For Chess with board size 8×8, it is $1\mathrm{e}50$
- For Checkers with board size 8×8, it is $1\mathrm{e}18$

[2] https://jku-vds-lab.at/publications/2020_tiis_pathexplorer/