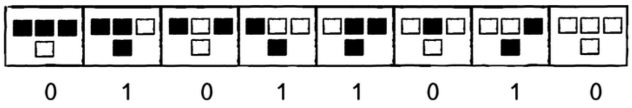
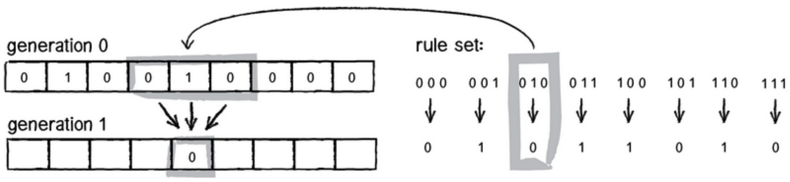
CA - discrete values

```
def ca():
   ''' Celluar automata with Python - K. Hong'''
   # 64 Boolean - True(1) : '*'
   #            - False(0): '-'
   # Rule - the status of current cell value is True
   # if only one of the two neighbors at the previous step is True('*')
   # otherwise, the current cell status is False('-')

   # list representing the current status of 64 cells
   ca = [
         0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,
         0,0,0,0,0,0,0,0,0,0, 0,1,0,0,0,0,0,0,0,0,
         0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,  0,0,0,0]

   # new Cellular values
   ca_new = ca[:]

   # dictionary maps the cell value to a symbol
   dic = {0:'-', 1:'*'}

   # initial draw - step 0
   print  ''.join( [dic[e] for e in ca_new])
   # additional 31 steps
   step = 1
   while(step < 32):
      ca_new = []
      # loop through 0 to 63 and store the current cell status in ca_new list
      for i in range(0,64):
         # inside cells - check the neighbor cell state
         if i > 0 and i < 63:
            if ca[i-1] == ca[i+1]:
               ca_new.append(0)
            else:
               ca_new.append(1)

         # left-most cell : check the second cell
         elif(i == 0):
            if ca[1] == 1:
               ca_new.append(1)
            else:
               ca_new.append(0)

         # right-most cell : check the second to the last cell
         elif(i == 63):
            if ca[62] == 1:
               ca_new.append(1)
            else:
               ca_new.append(0)

      # draw current cell state
      print  ''.join( [dic[e] for e in ca_new])

      # update cell list
      ca = ca_new[:]

      # step count
      step += 1

if __name__ == '__main__':
   ca()
```

So this particular rule can be illustrated as follows:





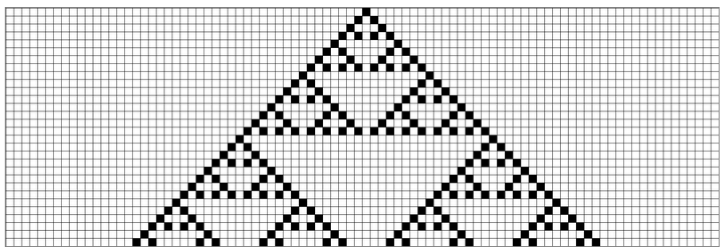


Figure 7.12: Rule 90

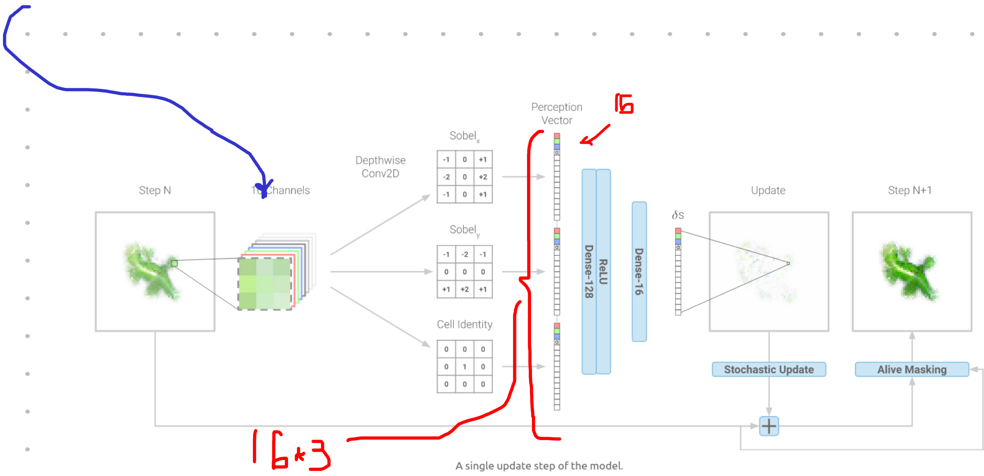
1. https://natureofcode.com/book/chapter-7-cellular-automata/
2. https://www.bogotobogo.com/python/python_cellular_automata.php

https://distill.pub/2020/growing-ca/

1. The biggest puzzle in this field is the question of how the cell collective knows what to build and when to stop

2. CAs typically consist of a grid of cells being iteratively updated, with the same set of rules being applied to each cell at every step.
The new state of a cell depends only on the states of the few cells in its immediate neighborhood. Despite their apparent simplicity,
CAs often demonstrate rich, interesting behaviours, and have a long history of being applied to modeling biological phenomena.

3. Usually, CA use discrete values for cell states.
Continious values - each cell is represented as a vector.
using continous. Therefore, we can use differentiable update rules.



A single update step of the model.

1. Each cell is a pixel with 16 channels [R,G,B,Alpha] Alpha > 0.1
    other values are "hidden states"

2. Each cell is allowed to look at its 8 neighbours across 16 channels

Apply Sobel filter x -
Apply Sobel filter y - gradients from left to right and from up to bottom

```
class Perception(nn.Module):
    def __init__(self, channels=16, norm_kernel=False):
        super().__init__()
        self.channels = channels
        sobel_x = torch.tensor([[-1.0, 0.0, 1.0],
                                [-2.0, 0.0, 2.0],
                                [-1.0, 0.0, 1.0]]) / 8
        sobel_y = torch.tensor([[1.0, 2.0, 1.0],
                                [0.0, 0.0, 0.0],
                                [-1.0, -2.0, -1.0]]) / 8
        identity = torch.tensor([[0.0, 0.0, 0.0],
                                 [0.0, 1.0, 0.0],
                                 [0.0, 0.0, 0.0]])

        self.kernel =  torch.stack((identity, sobel_x, sobel_y)).repeat(channels, 1, 1).unsqueeze(1)
        if norm_kernel:
            self.kernel /= channels

    def forward(self, state_grid):
        return F.conv2d(state_grid,
                        self.kernel.to(state_grid.device),
                        groups=self.channels,
                        padding=1)  # thanks https://github.com/PWhiddy/Growing-Neural-Cellular-Automat


class Policy(nn.Module):
```

https://github.com/belkakari/cellular-automata-pytorch/blob/master/modules/networks.py