GenCNN: Neural Network with 100% Accuracy for Cycle Recognization Task

Jiaming Shan

Shanghai Jiao Tong University

Shanghai, 2022

Content

1 Problem

2 Model Architecture

3 Discussion

Problem

If we generate a dataset which is easy for human to achieve 100% accuracy, can we get a model with 100% accuracy?
Probalbly not.

Cycle Dataset

- We build a cycle dataset.
- 60000 train images, 10000 test images.
- Each image has size 28x28, with every pixel 0 or 1.
- The label of a image is 0 or 1, 0 means there is no cycle, 1 means there is a cycle.

Here are some samples:

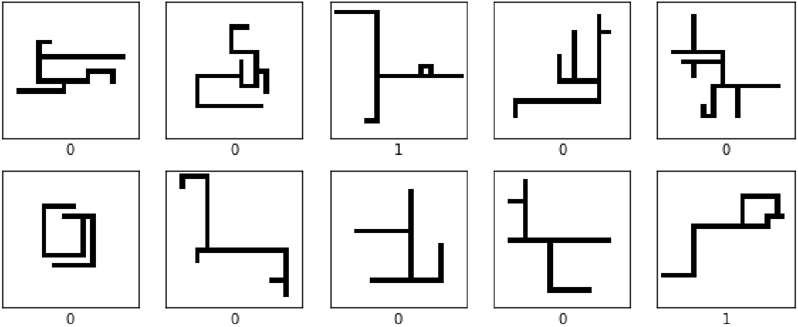

Figure: Cycle Dataset

- There exists deterministic algorithm to detect cycle.
- Can we build a deterministic neural network to detect cycle?

BFS Algorithm

We can use BFS in "1" pixel to solve this problem!

Algorithm 1

BFS. If go to a visited node, then there is a cycle.

Problem: This algorithm may return in every step. Difficult to implement in a neural network.

Algorithm 2

BFS to get the distance of every node to the starting point.
We can detect cycle easily by the distance image.

Easy to implement in neural network!
We will implement Algorithm 2.

Implement BFS Algorithm

Initializer:
Choose a random "1" pixel and set its distance to 0, and set all other pixels to $+\infty$.

Solver:
Input the distance data output by the initializer,
and do distance propagation operation repeatedly until all distance remains unchanged.

Extractor:
Detect whether there are patterns a pixel with distance d has at least two neighbor pixels with distance $d-1$.
If there are, output 1, showing that the image contains cycle, else output 0.

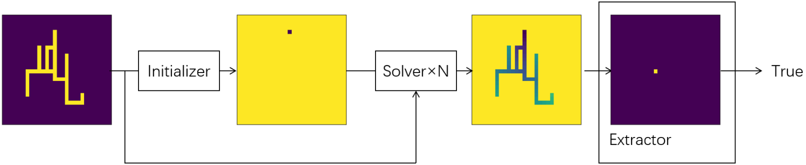

Figure: Data Flow of Deterministic Algorithm in Cycle Recognization

Solver Architecture

$$d_{merge} = padding(d + inputs \cdot \infty, 1, 1, +\infty)$$

$$\begin{aligned} d_{new}[i,j] = \min(& d_{merge}[i,j], \\ & d_{merge}[i+1,j]+1, \\ & d_{merge}[i-1,j]+1, \\ & d_{merge}[i,j+1]+1, \\ & d_{merge}[i,j-1]+1) \end{aligned}$$

Notice that we use the same raw inputs in each solver.
Call it global inputs.

Extractor Architecture

$$v = d[i, j]$$
$$l = [d[i, j-1], d[i-1, j], d[i+1, j], d[i, j+1]] - v$$
$$d[i, j] = sum(l == -1) >= 2$$

If any $d[i, j]$ outputs 1, then there is a cycle, and Extracor return 1, else return 0.

Turn BFS Algorithm to Neural Network

Solver and Extractor work like ConvNet!
Call it GenConv.
ConvNet: $x_{new} = \sigma(\Sigma w_i x_i + b)$. x_i is a neighborhood point of x.
GenConv: $x_{new} = f(X)$.
We call f GenKernel.
It is a more generalized CNN.
We call it GenCNN.

Build GenCNN

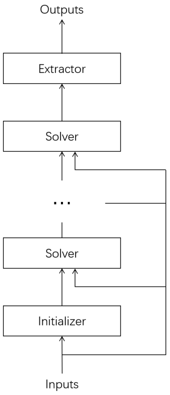

Figure: GenCNN Architecture

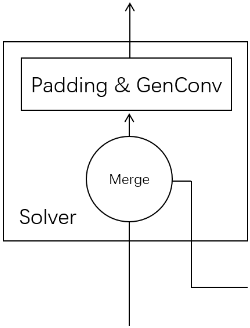

Figure: Solver Architecture

Build GenCNN

ReLU and Linear layer can do a lot of things!
Notice that

$$\min(a, b) = a - relu(a - b)$$

$$1_{a<b} = min(relu(b - a) * 10000, 1)$$

We can build GenKernel with a MLP with relu activation function.
We can use similiar tricks to build Extractor.

More Generalized GenCNN

A more generalized GenCNN has a preparer part to do features pre-extraction.
We need a good global inputs.
Also, $d[i, j]$ can be a vector instead of a number.

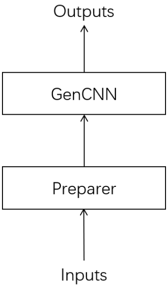

Figure: Generalized GenCNN Architecture

GenCNN vs CNN

Global Inputs

GenCNN has global inputs, like a deep ResNet, but works very differently.
CNN only uses the previous layer to predict the next layer.

GenKernel

GenCNN use GenKernel $x_{new} = f(X)$.
CNN use Kernel $x_{new} = \sigma(\Sigma w_i x_i + b)$.

Claim: GenCNN is similiar to GCN!

Advantages of Global Inputs

Global Inputs:
It will not be so smooth because we changed the fixed point.
Can be used to improve GCN and CNN!

Advantages of GenKernel

GenKernel:

- GenKernel can build all cellular automata!
- More Expressive!
- Wider inductive bias!
- More than image ...

Areas for Improvement

Depth

GenCNN is too Deep!
Reason: No dynamic loop structure in the network.

Speed

GenKernel is too slow if we use MLP.
Can we have a GenKernel different from Kernel but have the same speed and good result?

Areas for Improvement

Initializer

Initializer is not expressed by neural network in GenCNN for cycle dataset, but by hard logic.

Dataset

We need more harder deterministic dataset.

Questions?