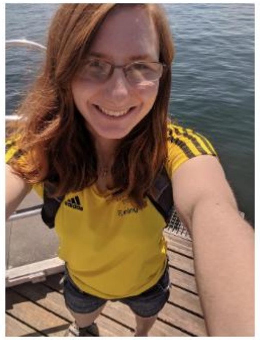


Ellen Arteca

PhD student at Northeastern University, working in programming languages research, with a focus on program analysis

论文汇报

《Learning How to Listen: Automatically Finding Bug Patterns in Event-Driven JavaScript APIs》	2022	TSE

Ellen Arteca, Max Schäfer, Frank Tip

《Visualization-Based Software Defect Prediction via Convolutional Neural Network with Global Self-attention》	2022	QRS

Shaojian Qiu[1,2], Shaosheng Wang[1], Xuhong Tian[1], Mengyang Huang[1], and Qiong Huang[1,2,*]

[1]College of Mathematics and Informatics, South China Agricultural University, Guangzhou, Guangdong, China

[2]Guangzhou Key Laboratory of Intelligent Agriculture, Guangzhou, Guangdong, China

qiushaojian@scau.edu.cn, qhuang@scau.edu.cn

*corresponding author

Shaojian Qiu

South China Agricultural University

Verified email at scau.edu.cn

Software Reliability Defect Prediction Transfer Learning

2023.5.4

Learning How to Listen: Automatically Finding Bug Patterns in Event-Driven JavaScript APIs

2022 TSE

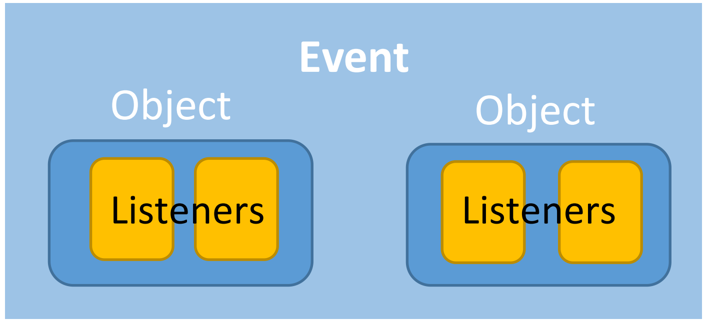
Motivation

If a listener registration misspells the name of the event or registers the listener on the wrong object, the listener will never be invoked. This is known as a ***dead listener*****.**

Wrong Time

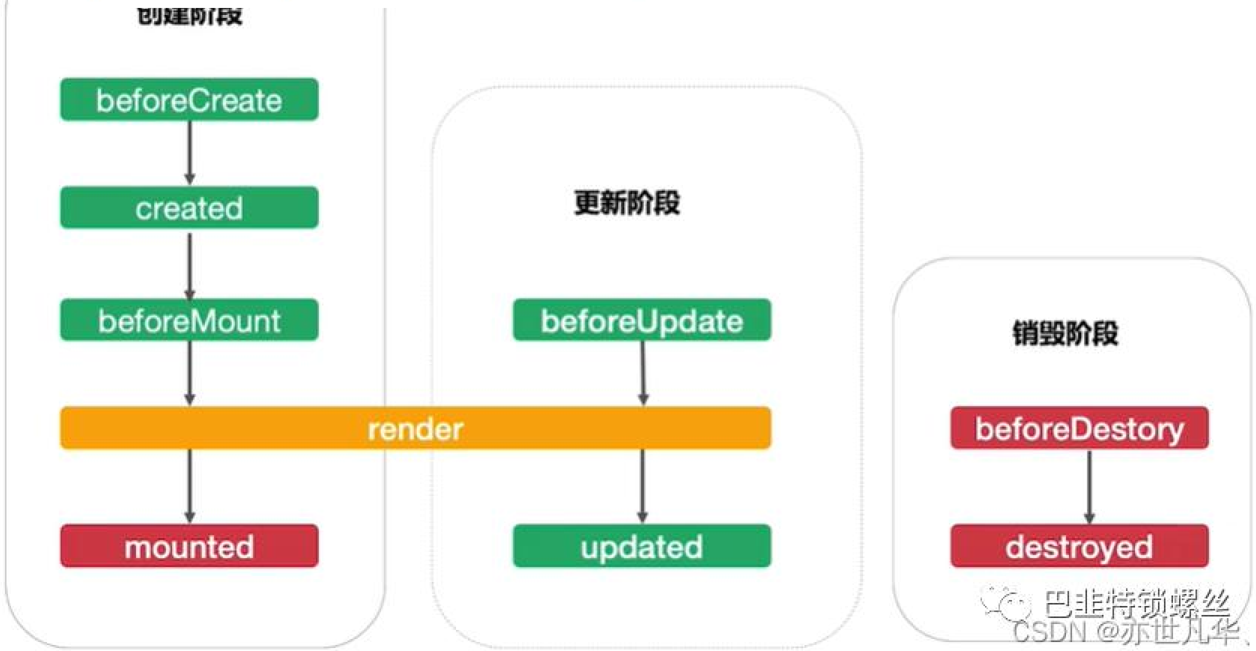
Misspell

Wrong Object




```
const http = require('http');
module.exports.request = (url) =>
  new Promise((resolve, reject) => {
    const req = http.request(url, res => {
      res.on('data', /* omitted */);
      res.on('end', () => {
        /* omitted */
        resolve( res);
      });
      res.on('timeout', () => reject(req)); // bug here
    });
    req.end();
  });
```

Fig. 2. An example of a dead-listener bug

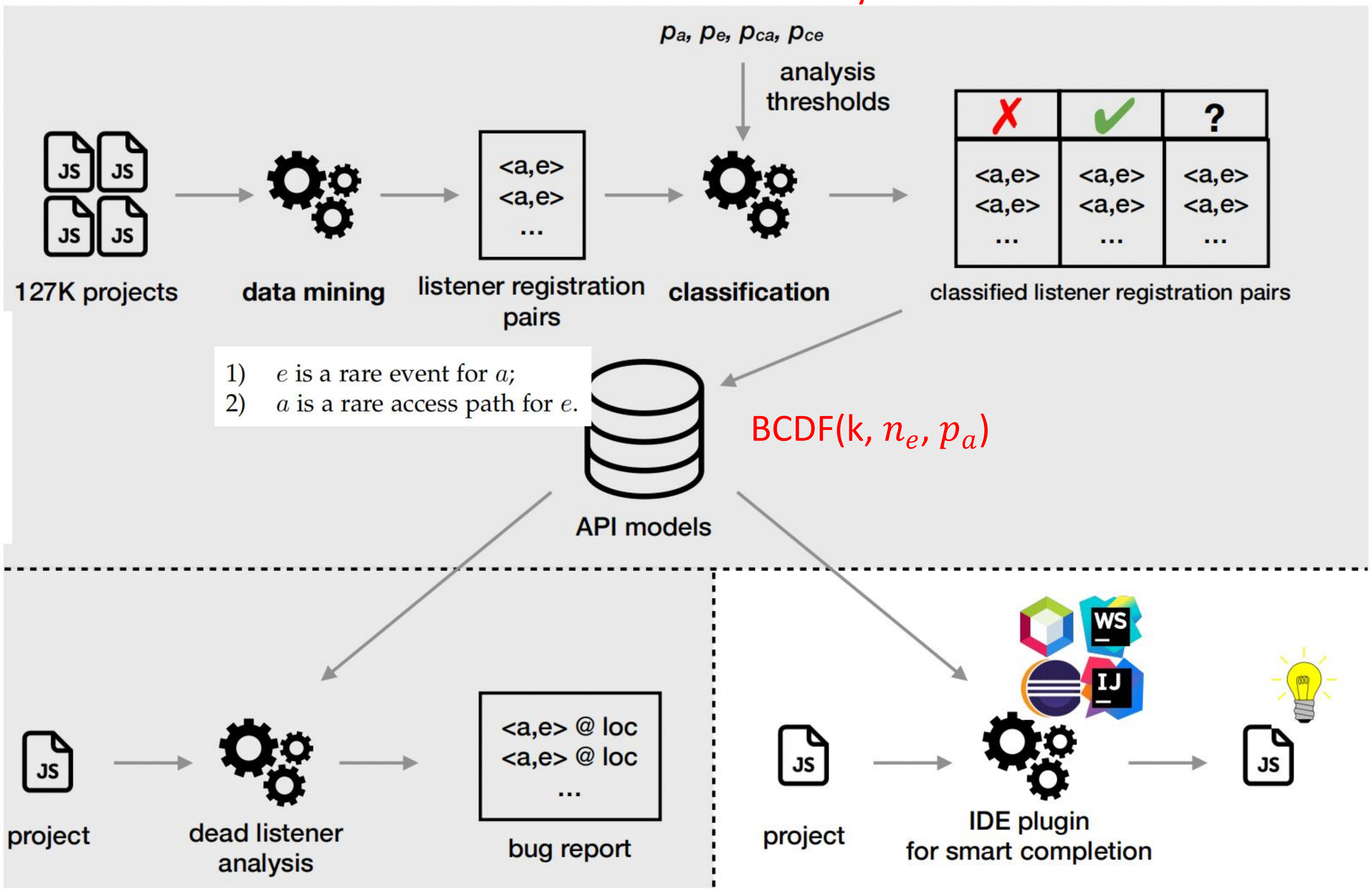


Learning How to Listen: Automatically Finding Bug Patterns in Event-Driven JavaScript APIs

2022 TSE

statistical analysis

Overview



1) $\langle \mathbf{require}(\texttt{http}).\texttt{request}(1)(0), \texttt{data}\rangle$, corresponds to line 5
2) $\langle \mathbf{require}(\texttt{http}).\texttt{request}(1)(0), \texttt{end}\rangle$, corresponds to line 6
3) $\langle \mathbf{require}(\texttt{http}).\texttt{request}(1)(0), \texttt{timeout}\rangle$, corresponds to line 10

$\langle \mathbf{require}(\texttt{http}).\texttt{request}(1)(0), \texttt{timeout}\rangle$

$n_e = 216$ and $k = 2$:

$\mathrm{BCDF}(2, 216, 0.05) \approx 0.001$

p_a / p_e: {0.005, 0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.25}

p_{ca} / p_{ce}: {0.005, 0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 1}

$\mathrm{BCDF}(k, n_a, p_e) < p_{ce} \ \wedge\ \mathrm{BCDF}(k, n_e, p_a) < p_{ca}$

Fig. 1. Overview of approach: the top half depicts the model-construction pipeline, while the bottom half shows their potential applications. This paper focuses on the shaded areas.

Visualization-Based Software Defect Prediction via Convolutional Neural Network with Global Self-attention

2022 QRS

Motivation

Intermediate Representation

Direct Program Expression

Channel attention information
Spatial attention informaiton

Local features extracting
Global relations among features

thread-unsafe

```
public class Clazz {
    private static DateFormat format
        = new SimpleDateFormat("yyyy/MM/dd");

    public Date method (String date) {
        return format.parse(date);
    }
}
```

Figure 1. A motivation example of a code snippet

Visualization-Based Software Defect Prediction via Convolutional Neural Network with Global Self-attention

2022 QRS

Framework

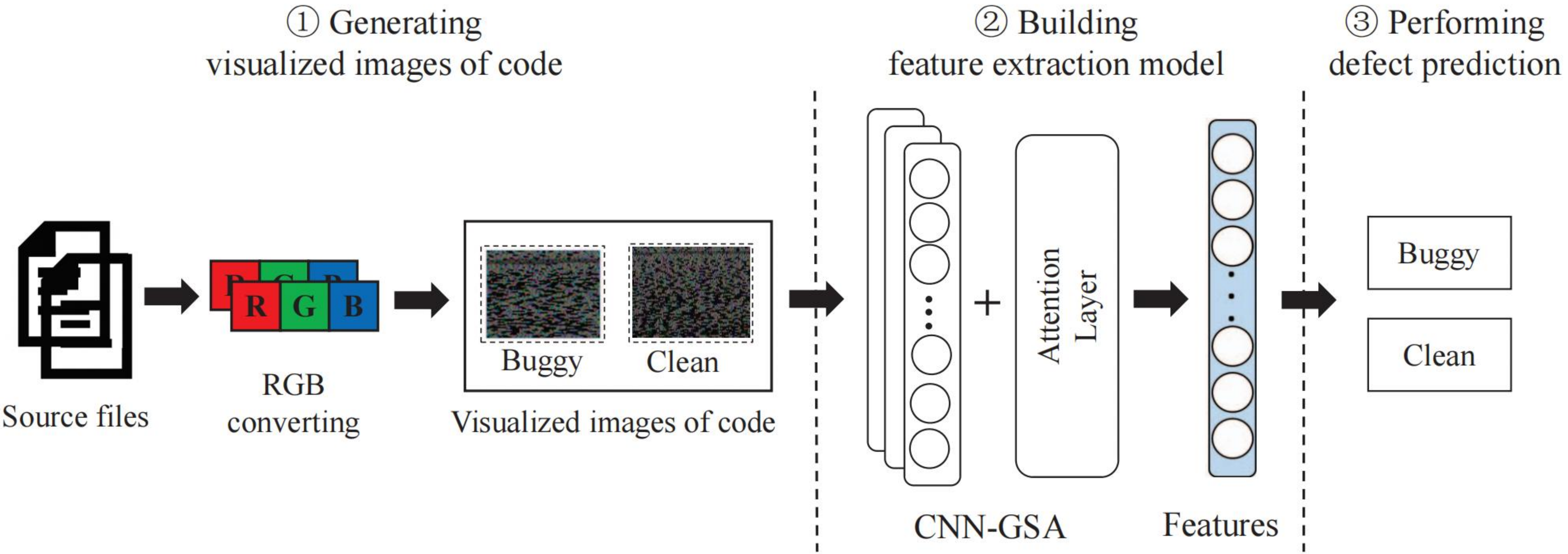


Figure 2. The Framework of the CNN-GSA

Visualization-Based Software Defect Prediction via Convolutional Neural Network with Global Self-attention

2022 QRS

public → ‘pub’ + ‘lic’ → [112,117,98] + [108,105,99]

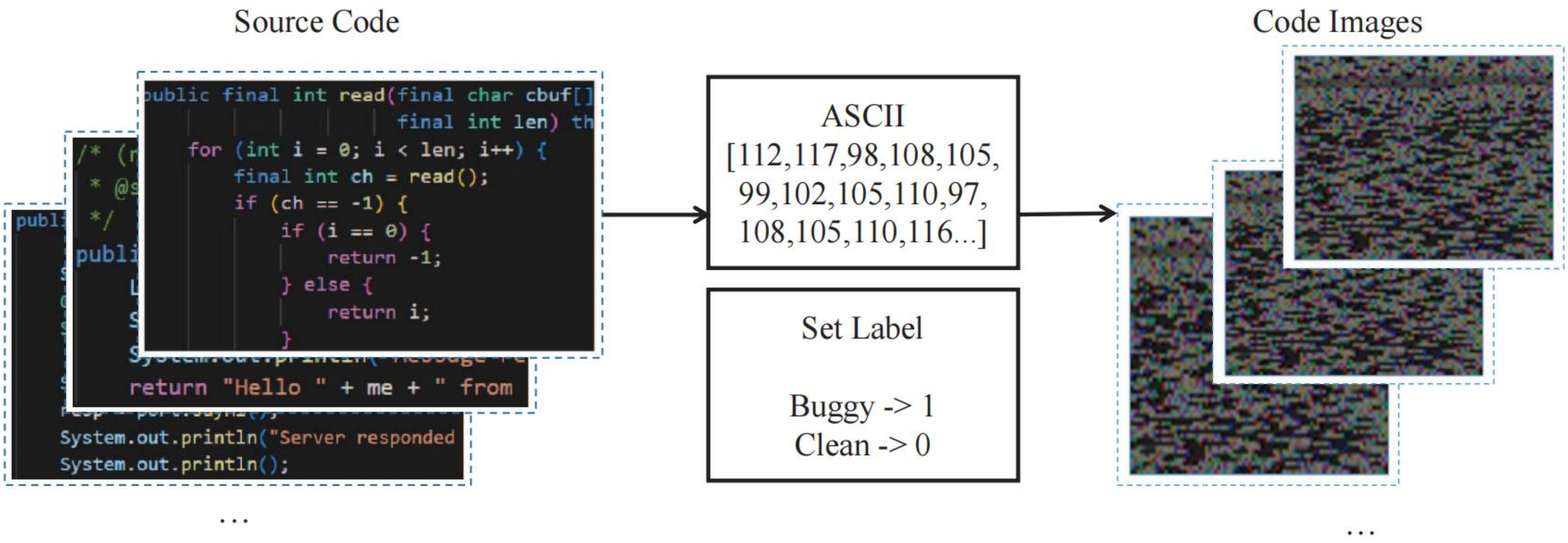


Figure 3. Converting code to RGB images

Vi al

RS

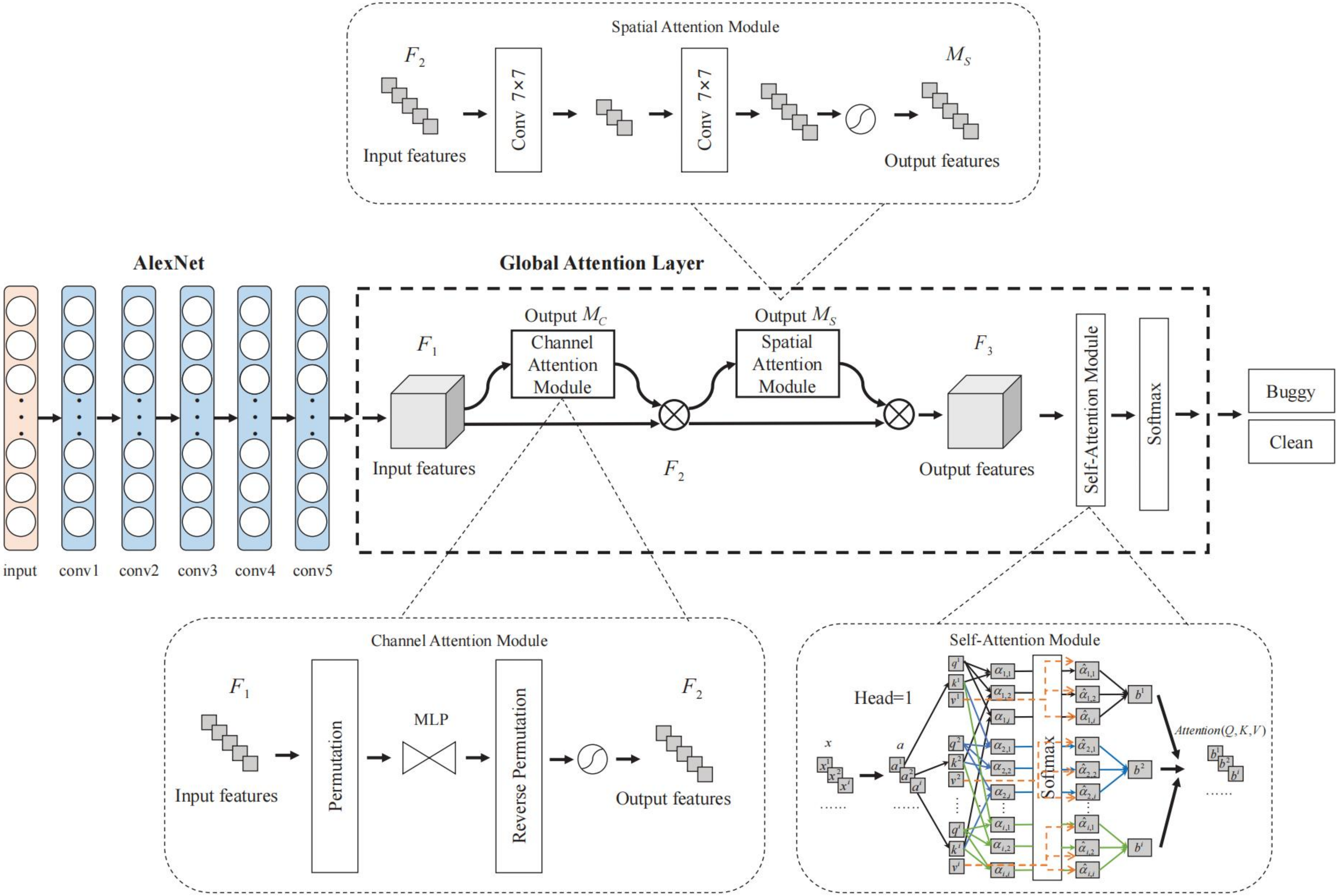


Figure 4. The network structure of CNN-GSA

谢谢聆听

演讲完毕