Reproduction Instructions for the Faster R-CNN

Wenjun Ma
Yunnan University
mwj22@stu.ynu.edu.cn

June 8, 2025

Abstract

This article provides an explanation of the second stage of the assessment. Here, I will elaborate on the initial design before the replication process, and describe my file organization structure and its contents. I will also introduce and analyze the problems I encountered during the replication process. Finally, I will summarize this replication and outline my next steps.

1 Introduction

The organizational structure of this explanation report is as follows: Section 2 introduces the preparations before the project implementation as well as the logic of code writing and testing. Section 3 presents the file structure of the project and briefly explains the functions of each file. Section 4 briefly introduces the related packages used in the project as well as the software and hardware environment. Section 5 elaborates and analyzes the problems I encountered. Section 6 provides a summary and next steps for this reproduction.

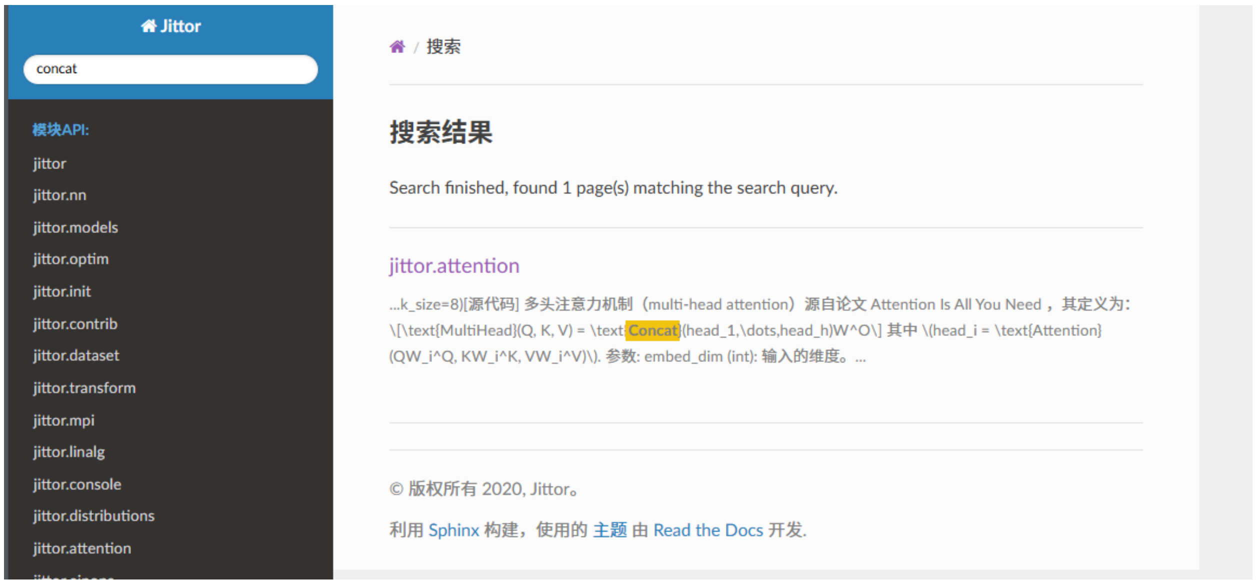
2 Initial design

Before starting the writing of this project, I decided to proceed in the following sequence: 1) Re-read the paper; 2) Read and study the relevant code (Faster R-CNN and Jittor); 3) Refer to the code and implement it by hand. Therefore, I found the following repositories and documents as the reference materials for my learning:

1. Jittor: a Just-in-time(JIT) deep learning framework
   `https://github.com/Jittor/jittor`
2. JRS
   `https://github.com/NK-JittorCV/nk-remote`
3. A Simple and Fast Implementation of Faster R-CNN
   `https://github.com/chenyuntc/simple-faster-rcnn-pytorch`
4. Documents of Jittor
   `https://cg.cs.tsinghua.edu.cn/jittor/assets/docs/index.html`
5. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks
   `https://github.com/shaoqingren/faster_rcnn`
6. Detectron2
   `https://github.com/facebookresearch/detectron2`

Among them, 1, 2 and 3 are the contents that I mainly referred to. Due to the absence of certain parts in the 4 (for example, `jt.concat` is not included in the document as shown in the Figure 1), 1 and 2 is basically used as a reference for the document.

The strategy for writing code For most of the code, Jittor is used for implementation (except when reading images, where `np.asarray` is still used in `TwoStageDectection/Utils/VOC_tools/data_utils.py`). In order to save the time for writing code and fully experience the usage of the Jittor framework, my code writing logic is as follows:



搜索结果

Search finished, found 1 page(s) matching the search query.

jittor.attention

...k_size=8)[源代码] 多头注意力机制（multi-head attention）源自论文 Attention Is All You Need，其定义为: \[\text{MultiHead}(Q, K, V) = \text{Concat}(head_1,\dots,head_h)W^O\] 其中 \(head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)\). 参数: embed_dim (int): 输入的维度。...

© 版权所有 2020, Jittor。

利用 Sphinx 构建，使用的 主题 由 Read the Docs 开发.

Figure 1: The result of the concat operation in the Jittor documentation search.

1. Directly invoke Jittor's API (such as `jt.ones`) or use existing code (such as `TwoStageDetection/Networks/Parts/RoiPool.py`.)
2. Although jittor has been integrated into the system, it is still recommended to use jittor for simple programming. For example, the network model can be implemented using jittor. For details, please refer to `TwoStageDectection/Networks/Parts/Backbone_VGG.py`.
3. Using `jt.code` + cuda for a simple implementation, as Listing 1

```
def nanmean(inputs):
    CUDA_HEADER = r'''
    #include <cmath>
    using namespace std;
    '''
    CUDA_SRC = r"""
        __global__ static void kernel1(@ARGS_DEF) {
            @PRECALC
            float sum = 0;
            int ing_sum = 0;
            for (int i = blockIdx.x * blockDim.x + threadIdx.x; i < in0_shape0;
            i += blockDim.x * gridDim.x){
                __isnanf(@in0(i)) ? ing_sum += 1 : sum += @in0(i);
            }
            int all_len = in0_shape0 - ing_sum;
            all_len == 0 ? @out(0) = NAN : @out(0) = sum / all_len;
        }
        const int total_count = in0_shape0;
        const int thread_per_block = 1024;
        const int block_count = (total_count + thread_per_block - 1) / thread_per_block;
        kernel1<<<block_count, thread_per_block>>>(@ARGS);
    """
    return jt.code(shape=(1,), dtype='float32', inputs=[inputs],
    cuda_header=CUDA_HEADER, cuda_src=CUDA_SRC)
```

Listing 1: A simple implementation of nanmean

The strategy for testing code The strategy of the code testing is relatively simpler. Some independent arithmetic operations are tested separately, while those strongly related to model training and testing are checked simultaneously during the overall operation. (When considering this, I thought the overall project was not difficult, so I made this somewhat hasty decision.)

3 File structure

This section presents the file structure of the entire project and briefly describes the contents of each file.

```
TwoStageDectection
├─ Log
├─ Main
│  ├─ Main.py ........................ The main running file.
│  └─ PreConfig.ini .................. Configuration file.
├─ Networks
│  ├─ Parts
│  │  ├─ Backbone_Resnet.py .......... The basic implementation of ResNet.
│  │  ├─ Backbone_VGG.py ............. The basic implementation of VGG.
│  │  ├─ RoiPool.py .................. RoiPool is derived from JRS.
│  │  └─ RPN.py ...................... The basic implementation of RPN.
│  ├─ Base_FasterRCNN.py ............. The basic implementation of Faste rR-CNN.
│  └─ VGG16_FasterRCNN.py ............ The implementation of Faster R-CNN based on VGG16.
├─ Utils
│  ├─ Base
│  │  ├─ BBox.py ..................... The basic operations of BBox.
│  │  ├─ Creator.py .................. Tools for Region Generation and Region Calculation.
│  │  ├─ eval_tools.py ............... Tools for calculating the evaluation indicators.
│  │  ├─ Losses.py ................... The definition of the loss function.
│  │  └─ Trainer.py .................. The trainer used for training the model.
│  └─ VOC_tools
│     ├─ data_utils.py ............... Tools for processing data.
│     └─ VOCDataset.py ............... Tools for reading the VOC dataset
├─ requirements.txt
└─ test.py
```

4 Environment configuration

This section presents the environment in which the project operates and the logic for the model's operation.

4.1 Environment configuration

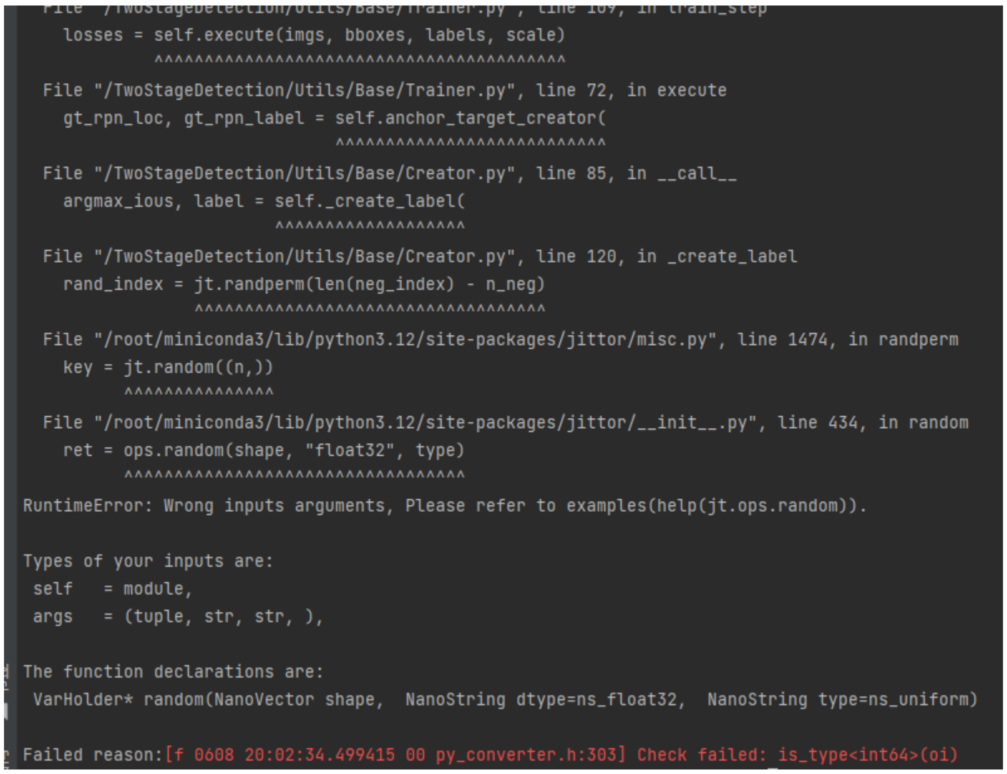
Because the Conda environment is rather messy, we did not directly generate environment.yaml. Instead, we used pipreqs to generate requirements.txt as shown below.

```
jittor==1.3.9.14
numpy==2.3.0
Pillow==11.2.1
tqdm==4.66.2
```

Listing 2: requirements.txt

System environment and hardware:

- CUDA 12.4
- GPU RTX4090D
- ubuntu22.04



```
  File "/TwoStageDetection/Utils/Base/Trainer.py", line 109, in train_step
    losses = self.execute(imgs, bboxes, labels, scale)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/TwoStageDetection/Utils/Base/Trainer.py", line 72, in execute
    gt_rpn_loc, gt_rpn_label = self.anchor_target_creator(
                               ^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/TwoStageDetection/Utils/Base/Creator.py", line 85, in __call__
    argmax_ious, label = self._create_label(
                         ^^^^^^^^^^^^^^^^^^^
  File "/TwoStageDetection/Utils/Base/Creator.py", line 120, in _create_label
    rand_index = jt.randperm(len(neg_index) - n_neg)
                 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/root/miniconda3/lib/python3.12/site-packages/jittor/misc.py", line 1474, in randperm
    key = jt.random((n,))
          ^^^^^^^^^^^^^^^
  File "/root/miniconda3/lib/python3.12/site-packages/jittor/__init__.py", line 434, in random
    ret = ops.random(shape, "float32", type)
          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
RuntimeError: Wrong inputs arguments, Please refer to examples(help(jt.ops.random)).

Types of your inputs are:
 self   = module,
 args   = (tuple, str, str, ),

The function declarations are:
 VarHolder* random(NanoVector shape,  NanoString dtype=ns_float32,  NanoString type=ns_uniform)

Failed reason:[f 0608 20:02:34.499415 00 py_converter.h:303] Check failed: is_type<int64>(oi)
```

Figure 2: Error message0.

5 The problems encountered and the solutions.

This section mainly briefly discusses the issues related to environment configuration and focuses on the problems of the code.

5.1 The problems of environment configuration

Due to reasons such as the sudden failure of the graphics card on the laboratory host computer, or the sudden malfunction of the motherboard, I configured many devices. Eventually, I ran the code on a cloud server. The main issues with the environment configuration can be divided into two types: one is the absence of files, and the other is other problems.

5.1.1 Absence of files

The handling method for missing files is quite simple. If a certain file is missing, simply use the command `find -name` to search for it, and then use `ln -s` to create a link.

5.1.2 Other problems

Based on the experience I have gained from issues and the community, I will handle problems in the following three steps:

1. Constantly try out different versions of Jittor. (Successfully configured the environment on a certain Windows system using version 1.3.6.6.)

2. Constantly try out different versions of CUDA.

3. Reinstall the operating system.

5.2 The problems of code

This section presents some of the problems I encountered. For each problem, I will provide a wrong example, my provisional solution, the current situation of the problem and the analysis process, and I haven't had the chance to delve into the content of the source code that is built using C yet, so the source code only shows the part implemented in Python.

In order to make the problem occur more consistently, the data shuffling has been disabled here.

5.2.1 problem0

Example & (provisional) Solution This issue originates from line 120 of `TwoStageDectection/Utils/Base/Creator.py`. The error code and the corrected code are shown as follows.

```
wrong code:
    rand_index = jt.randperm(len(neg_index) - n_neg)
right code:
    rand_index = jt.randperm(len(neg_index) - n_neg.item())
```

Listing 3: problem0

The error message is shown in Figure 3.

Reproducibility True

Problem Analysis This issue is named in this way because it was originally functional, but suddenly it stopped working. Now, let's analyze this situation. First, I viewed the source code of the relevant function, shown as Listing 4.

```
def randperm(n, dtype="int32"):
    key = jt.random((n,))
    index, _ = jt.argsort(key)
    return index.cast(dtype)

def random(shape, dtype="float32", type="uniform"):
     for dim in shape:
            if dim < 0:
                raise RuntimeError(f"Trying to create tensor with \
                negative dimension {dim}: {shape}")
        ret = ops.random(shape, "float32", type)
        amp_reg = jt.flags.amp_reg
        if amp_reg:
            if amp_reg & 16:
                if amp_reg & 1:
                    if ret.dtype != "float32":
                        return ret.float32()
                elif amp_reg & 2:
                    if ret.dtype != "float16":
                        return ret.float16()
        return ret

def subtract(self, y: Var)-> Var:
```

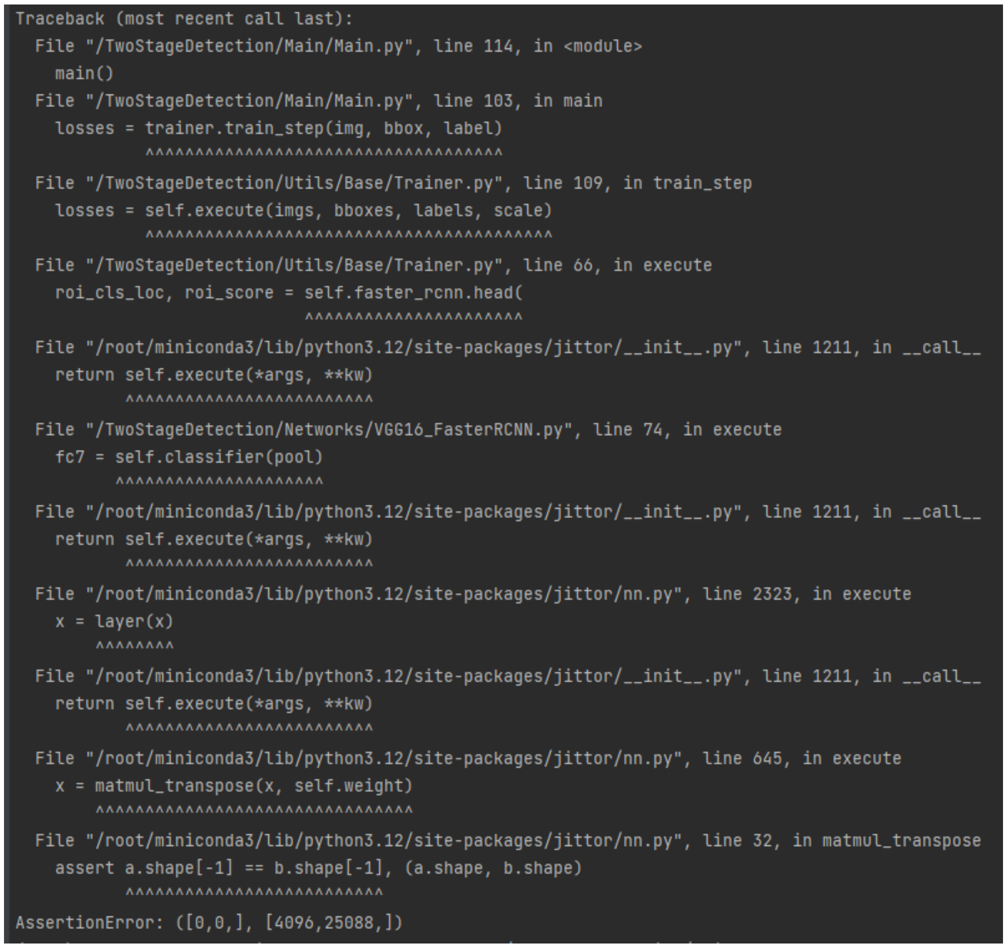
Listing 4: source0

Then we will print out the inputs for both scenarios, shown as Listing 5.

```
wrong code:
    AssertionError: jt.Var([24], dtype=int32)
right code:
    AssertionError: 277
```

Listing 5: ptintout0

We can observe that the output of both methods is of the `int` type. According to the error message, the type of `jt.Var` is also `int32`. Referring to the error information, there is an `int64` that does not match the basic Python type `int`. This



```
Traceback (most recent call last):
  File "/TwoStageDetection/Main/Main.py", line 114, in <module>
    main()
  File "/TwoStageDetection/Main/Main.py", line 103, in main
    losses = trainer.train_step(img, bbox, label)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/TwoStageDetection/Utils/Base/Trainer.py", line 109, in train_step
    losses = self.execute(imgs, bboxes, labels, scale)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/TwoStageDetection/Utils/Base/Trainer.py", line 66, in execute
    roi_cls_loc, roi_score = self.faster_rcnn.head(
                             ^^^^^^^^^^^^^^^^^^^^^^
  File "/root/miniconda3/lib/python3.12/site-packages/jittor/__init__.py", line 1211, in __call__
    return self.execute(*args, **kw)
           ^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/TwoStageDetection/Networks/VGG16_FasterRCNN.py", line 74, in execute
    fc7 = self.classifier(pool)
          ^^^^^^^^^^^^^^^^^^^^^
  File "/root/miniconda3/lib/python3.12/site-packages/jittor/__init__.py", line 1211, in __call__
    return self.execute(*args, **kw)
           ^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/root/miniconda3/lib/python3.12/site-packages/jittor/nn.py", line 2323, in execute
    x = layer(x)
        ^^^^^^^^
  File "/root/miniconda3/lib/python3.12/site-packages/jittor/__init__.py", line 1211, in __call__
    return self.execute(*args, **kw)
           ^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/root/miniconda3/lib/python3.12/site-packages/jittor/nn.py", line 645, in execute
    x = matmul_transpose(x, self.weight)
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/root/miniconda3/lib/python3.12/site-packages/jittor/nn.py", line 32, in matmul_transpose
    assert a.shape[-1] == b.shape[-1], (a.shape, b.shape)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: ([0,0,], [4096,25088,])
```

Figure 3: Error message1_0.

error may occur because the source code failed to correctly read the data of var here, and instead mistakenly converted the entire type into a type similar to `np.int64` . Another supporting basis for this inference is that the dependency package of jittor contains numpy, which may have performed related operations.

5.2.2 problem1

Example & (provisional) Solution This issue has not been resolved yet. Here, only the problem description and error information are provided. This problem is also the biggest obstacle I encountered during the implementation process.

Reproducibility True

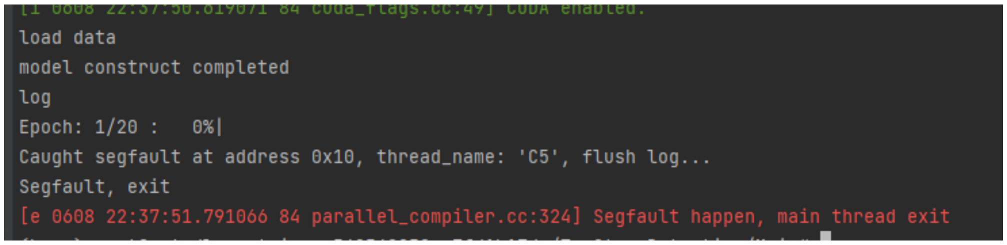
Problem Analysis The problem directly related to the error message is caused by `TwoStageDectection/Networks/VGG16_FasterRCNN.py`. There is no data in the input pool. After conducting a thorough investigation, I found that this step shown as Listing 6 in line 202 of `TwoStageDectection/Utils/Base/Creator.py` resulted in the data being empty. There were no data that met the filtering requirements.

```
keep = jt.where((hs >= min_size) & (ws >= min_size))[0]
```

Listing 6: code1

When data shuffling was enabled, I printed the contents of the data multiple times. On one occasion, I discovered that there were `nan` values in the data.

Regarding the methods for handling `nan` values in the data, I made the following attempts for processing, but all failed.



```
[I 0608 22:37:50.819071 84 cuda_flags.cc:49] CUDA enabled.
load data
model construct completed
log
Epoch: 1/20 :   0%|
Caught segfault at address 0x10, thread_name: 'C5', flush log...
Segfault, exit
[e 0608 22:37:51.791066 84 parallel_compiler.cc:324] Segfault happen, main thread exit
```

Figure 4: Error message1_1.

1. Modified the initialization method of the convolution layer.
2. Adjusted the learning rate.
3. Applied gradient clipping.
4. Disabled data shuffling and swapped or deleted the data.
5. Limited and adjusted the numerical range of all bounding box calculation values.
6. Used the official `export JT_CHECK_NAN=1, export trace_py_var=3` provided by the Jittor to locate the error position.

Based on this, I plan to use `assert not jt.isnan(features).any(), "!!!"` to locate and find out exactly where the problem lies. However, a segmentation fault occurred, as shown in the Figure 4.

The official debugging method(`export debug=1`, `export gdb_attach=1`) provided by the Jittor was used to detect the segmentation fault, then the investigation was carried out. Once before, there was a prompt indicating that the accessed memory was already released, but no screenshot was taken at that time. From now on, if we test again, the only possible outcomes would be either a system crash or no additional error messages.

Based on this, I can infer that there is a problem with Jittor's memory management. It is very likely that there is an issue with the counting or marking during garbage collection, which leads to the incorrect release of memory.

6 Conclusion