TroubleShooter: MindSpore 网络开发调试工具包

分享人：樊大为

昇思 MindSpore

TroubleShooter

网络迁移 (migrator)

- pth权重迁移
- ckpt校验
- 数据保存/比较
- 网络正向自动输出比较

故障分析 (proposer)

- 在线异常建议
- 离线报错分析
- 简洁调用栈

跟踪调试 (tracker)

- 网络运行信息
- 获取INF/NAN值抛出点
- 黑白名单过滤追踪信息
- 跟踪API报错详细执行过程

Gitee: https://gitee.com/MindSpore/toolkits/tree/master/troubleshooter

网络迁移 (migrator)

- pth权重迁移
- ckpt校验
- 数据保存/比较
- 网络正向自动输出比较

torch权重自动化迁移

从PyTorch网络迁移到MindSpore网络时，两者的ckpt文件的编码不同，无法直接加载，需要权重转换。

MindSpore的API实现细节上与PyTorch不同，权重名称与PyTorch有差异（例如norm层中的weight和gamma等）。

为提高网络迁移易用性，troubleshooter提供了完备的权重转换工具。

torch权重自动化迁移

网络结构完全一致，权重自动转换

```
import troubleshooter as ts
```

```
2023-01-04 14:50:57,529 - troubleshooter.log - WARNING - [*User Attention*] The PTH has been converted to the checkpoint of MindSpore. Please check whether the conversion result is correct. The saved path is: ./convert_resnet.ckpt
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
|                                                                The list of conversion result                                                                          |
+---------------------------------------+---------------------------------------+-------------------------------+-------------------------------+
|          Parameter name of pth        |     Parameter name of converted ckpt  | Whether the name is converted |    Parameter shape of pth     |
+---------------------------------------+---------------------------------------+-------------------------------+-------------------------------+
|              conv1.weight             |              conv1.weight             |             False             |  torch.Size([64, 3, 7, 7])    |
|               bn1.weight              |               bn1.gamma               |              True             |       torch.Size([64])        |
|                bn1.bias               |                bn1.beta               |              True             |       torch.Size([64])        |
|           bn1.running_mean            |           bn1.moving_mean             |              True             |       torch.Size([64])        |
|            bn1.running_var            |         bn1.moving_variance           |              True             |       torch.Size([64])        |
|       bn1.num_batches_tracked         |       bn1.num_batches_tracked         |             False             |        torch.Size([])         |
|        layer1.0.conv1.weight          |        layer1.0.conv1.weight          |             False             |  torch.Size([64, 64, 1, 1])   |
|         layer1.0.bn1.weight           |          layer1.0.bn1.gamma           |              True             |       torch.Size([64])        |
|          layer1.0.bn1.bias            |          layer1.0.bn1.beta            |              True             |       torch.Size([64])        |
|      layer1.0.bn1.running_mean        |      layer1.0.bn1.moving_mean         |              True             |       torch.Size([64])        |
|       layer1.0.bn1.running_var        |     layer1.0.bn1.moving_variance      |              True             |       torch.Size([64])        |
|    layer1.0.bn1.num_batches_tracked   |    layer1.0.bn1.num_batches_tracked   |             False             |        torch.Size([])         |
|        layer1.0.conv2.weight          |        layer1.0.conv2.weight          |             False             |  torch.Size([64, 64, 3, 3])   |
|         layer1.0.bn2.weight           |          layer1.0.bn2.gamma           |              True             |       torch.Size([64])        |
|          layer1.0.bn2.bias            |          layer1.0.bn2.beta            |              True             |       torch.Size([64])        |
|      layer1.0.bn2.running_mean        |      layer1.0.bn2.moving_mean         |              True             |       torch.Size([64])        |
|       layer1.0.bn2.running_var        |     layer1.0.bn2.moving_variance      |              True             |       torch.Size([64])        |
|    layer1.0.bn2.num_batches_tracked   |    layer1.0.bn2.num_batches_tracked   |             False             |        torch.Size([])         |
|        layer1.0.conv3.weight          |        layer1.0.conv3.weight          |             False             |  torch.Size([256, 64, 1, 1])  |
|         layer1.0.bn3.weight           |          layer1.0.bn3.gamma           |              True             |      torch.Size([256])        |
```

```
ts.migrator.convert_weight(weight_map_path="/tmp/torch_net_map.json",
                           pt_file_path="/tmp/torch_net.pth",
                           ms_file_save_path='/tmp/convert_resnet.ckpt')
```

torch权重自动化迁移

网络结构有一定差异，需要定制权重名称前缀

```
import troubleshooter as ts

# pytorch的resnet50网络
torch_net = resnet50(num_classes=10)
pth_path="./resnet.pth"

# weight_name_prefix：需要添加的权重前缀
ts.migrator.get_weight_map(pt_model=torch_net,
                           weight_map_save_path="/tmp/torch_net_map.json",
                           weight_name_prefix='uvp',
                           print_map=True)
# 调用转换接口
ts.migrator.convert_weight(weight_map_path="/tmp/torch_net_map.json",
                           pt_file_path="/tmp/torch_net.pth",
                           ms_file_save_path='/tmp/convert_resnet.ckpt')
```

torch权重自动化迁移

网络结构有一定差异，需要对权重名称做复杂的定制转换

```python
import troubleshooter as ts

def custorm_weight_name(weight_name_map):
    prefix='.custorm.'
    custorm_name_map = {}
    for key, value in weight_name_map.items():
        index = value.find(".")
        value = value[0:index] + prefix + value[index+1:]
        print(key, ":", value)
        custorm_name_map[key] = str(value)
    return custorm_name_map
```

获得根据默认规则转换后的map，之后执行自定义函数，进行自定义的修改。

```python
# pytorch的resnet50网络
torch_net = resnet50(num_classes=10)
pth_path="./resnet.pth"

"""
custom_name_func: 可封装定制函数，例如：custorm_weight_name，完成映射关系的定制
"""
ts.migrator.get_weight_map(pt_model=torch_net,
                           weight_map_save_path="/tmp/torch_net_map.json",
                           custom_name_func=custorm_weight_name,
                           print_map=True)

# 调用转换接口
ts.migrator.convert_weight(weight_map_path="/tmp/torch_net_map.json",
                           pt_file_path="/tmp/torch_net.pth",
                           ms_file_save_path='/tmp/convert_resnet.ckpt')
```

执行结果：根据定制所有参数名称增加一个层custorm ，例如：
features.Linear_mm.weight 参数名称将转换为
features.custorm.Linear_mm.weight

转换后的ckpt与MindSpore比较校验

将网络生产的*ckpt*与转换的*ckpt*（用权重转换工具从*pth*转换过来的*ckpt*）进行对

The list of comparison results

Parameter name of input ckpt	Parameter name of converted ckpt	Whether shape are equal	Parameter shape of input ckpt	Parameter shape of converted ckpt
conv1.weight	conv1.weight	True	(64, 3, 7, 7)	(64, 3, 7, 7)
bn1.moving_mean	bn1.moving_mean	True	(64,)	(64,)
bn1.moving_variance	bn1.moving_variance	True	(64,)	(64,)
bn1.gamma	bn1.gamma	True	(64,)	(64,)
bn1.beta	bn1.beta	True	(64,)	(64,)
layer1.0.conv1.weight	layer1.0.conv1.weight	True	(64, 64, 1, 1)	(64, 64, 1, 1)
layer1.0.bn1.moving_mean	layer1.0.bn1.moving_mean	True	(64,)	(64,)
layer1.0.bn1.moving_variance	layer1.0.bn1.moving_variance	True	(64,)	(64,)
layer1.0.bn1.gamma	layer1.0.bn1.gamma	True	(64,)	(64,)
layer1.0.bn1.beta	layer1.0.bn1.beta	True	(64,)	(64,)
layer1.0.conv2.weight	layer1.0.conv2.weight	True	(64, 64, 3, 3)	(64, 64, 3, 3)
layer1.0.bn2.moving_mean	layer1.0.bn2.moving_mean	True	(64,)	(64,)
layer1.0.bn2.moving_variance	layer1.0.bn2.moving_variance	True	(64,)	(64,)
layer1.0.bn2.gamma	layer1.0.bn2.gamma	True	(64,)	(64,)
layer1.0.bn2.beta	layer1.0.bn2.beta	True	(64,)	(64,)
layer1.0.conv3.weight	layer1.0.conv3.weight	True	(256, 64, 1, 1)	(256, 64, 1, 1)
layer1.0.bn3.moving_mean	layer1.0.bn3.moving_mean	True	(256,)	(256,)
layer1.0.bn3.moving_variance	layer1.0.bn3.moving_variance	True	(256,)	(256,)
layer1.0.bn3.gamma	layer1.0.bn3.gamma	True	(256,)	(256,)
layer1.0.bn3.beta	layer1.0.bn3.beta	True	(256,)	(256,)
layer1.0.downsample.0.weight	layer1.0.downsample.0.weight	True	(256, 64, 1, 1)	(256, 64, 1, 1)

```
ts.migrator.compare_pth_and_ckpt(weight_map_path, pth_path, ms_path)
```

统一的数据保存接口

在网络迁移精度问题排查时，需要对网络中的数据进行保存。

原始方案：

1. 静态图中在网络中使用print（实际调用ops.Print()算子），对tensor进行打印

   难以对大量数据进行输出，难以保存供后续对比使用。

2. 使用TensorSummary接口保存到summary文件

   必须和SummaryRecord或SummaryCollector一起使用，学习成本高。

统一的数据保存接口

troubleshooter提供了支持MindSpore和PyTorch统一的数据保存接口

ts.save

可以将tensor存储数据为numpy，方便后续统一比较。（适配2.0正式版）

易用性：

1. 支持PyTorch和MindSpore（pynative/图模式），自动识别数据类型。

2. 支持保存tensor，list[tensor]/tuple[tensor]/dict[str, tensor]

3. 支持自动编号功能，多个迭代文件不覆盖。

统一的数据保存接口

接口定义

```
ts.save(file:str, data:Union(Tensor, list[Tensor], tuple[Tensor], dict[str, Tensor], auto_id=True, suffix=None))
```

- file: 文件名路径。当 `file` 为 `None` 或 `''` 时，文件名会自动设置为 `tensor_(shape)`，文件路径为当前路径。
- data: 数据，支持保存 `Tensor` （包括 `mindspore.Tensor` 和 `pytorch.tensor` ），以及 `Tensor` 构成的 `list/tuple/dict` 。当为 `list/tuple` 类型时，会按照顺序添加编号；当为 `dict` 类型时，文件名中会添加 `key` 。
- auto_id: 自动编号，默认值为 `True` 。当为 `True` 时，保存时会自动为文件添加全局编号，编号从0开始。
- suffix: 文件名后缀，默认值为 `None` 。

文件保存格式

存储的文件名称为 `{id}_name_{idx/key}_{suffix}.npy`

统一的数据保存接口

MindSpore

```
import os
import shutil

import troubleshooter as ts
import mindspore as ms
from mindspore import nn, Tensor

class NetWorkSave(nn.Cell):
    def __init__(self, file):
        super(NetWorkSave, self).__init__()
        self.file = file

    def construct(self, x):
        ts.save(self.file, x)
        return x

x1 = Tensor(-0.5962, ms.float32)
x2 = Tensor(0.4985, ms.float32)
try:
    shutil.rmtree("/tmp/save/")
except FileNotFoundError:
    pass
os.makedirs("/tmp/save/")
net = NetWorkSave('/tmp/save/ms_tensor')
```

```
x1 = Tensor(-0.5962, ms.float32)
x2 = Tensor(0.4985, ms.float32)
try:
    shutil.rmtree("/tmp/save/")
except FileNotFoundError:
    pass
os.makedirs("/tmp/save/")
net = NetWorkSave('/tmp/save/ms_tensor')

# 支持自动编号
out1 = net(x1)
# /tmp/save/0_ms_tensor.npy

out2 = net(x2)
# /tmp/save/1_ms_tensor.npy

out3 = net([x1, x2])
# /tmp/save/2_ms_tensor_0.npy
# /tmp/save/2_ms_tensor_1.npy

out4 = net({"x1": x1, "x2":x2})
# /tmp/save/3_ms_tensor_x1.npy
# /tmp/save/3_ms_tensor_x2.npy
```

支持自动编号

支持tensor/list/dict多种数据格式

统一的数据保存接口

PyTorch

```python
import os
import shutil

import troubleshooter as ts
import torch
x1 = torch.tensor([-0.5962, 0.3123], dtype=torch.float32)
x2 = torch.tensor([[0.4985],[0.4323]], dtype=torch.float32)

try:
    shutil.rmtree("/tmp/save/")
except FileNotFoundError:
    pass
os.makedirs("/tmp/save/")

file = '/tmp/save/torch_tensor'
```

```python
ts.save(file, x1)
# /tmp/save/0_torch_tensor.npy
ts.save(file, x2)
# /tmp/save/1_torch_tensor.npy
ts.save(None, {"x1":x1, "x2":x2}, suffix="torch")
# ./2_tensor_(2,)_x1_torch.npy
# ./2_tensor_(2, 1)_x2_torch.npy
```

当file为空时，会为文件自动命名为

tensor_{data.shape}

数据比对接口

进行网络迁移精度问题排查等场景，需要获取网络中的tensor值进行比较。

一般我们将tensor保存成npy进行手工比较，此功能提供了批量对比两个目录下名称可以映射的npy值的接口。

数据比对接口

直接映射

```
import troubleshooter as ts

ta = "/mnt/d/06_project/troubleshooter/troubleshooter/tests/diff_handler/ta"
tb = "/mnt/d/06_project/troubleshooter/troubleshooter/tests/diff_handler/tb"
# 比较ta与tb目录下名称可以映射的npy文件的值
ts.migrator.compare_npy_dir(ta, tb)
```

```
The orig dir: /tmp/troubleshooter_ta/
The target dir: /tmp/troubleshooter_tb/
+-----------------------------------------------------------------------------------------------------------------------------------+
|                                                The list of comparison results                                                     |
+-----------------+--------------------+-----------------------+-------------+--------------------+-----------------------------------+
| orig array name | target array name  | results of comparison | match ratio | cosine similarity  |          (mean, max, min)         |
+-----------------+--------------------+-----------------------+-------------+--------------------+-----------------------------------+
|    data1.npy    |     data1.npy      |          True         |   100.00    |      1.00000       |                 []                |
|    data2.npy    |     data2.npy      |         False         |    83.33    |      0.99924       | ['0.013382', '0.080290', '0.000000'] |
+-----------------+--------------------+-----------------------+-------------+--------------------+-----------------------------------+
```

ta　　tb　　allclose结果　　allclose的正确率　　余弦相似度　　差异值的指标

allclose成功数/总数量

数据比对接口

两个目录下名称不能完全映射，需要手工调整

```
import troubleshooter as ts

ta = "/mnt/d/06_project/troubleshooter/troubleshooter/tests/diff_handler/ta"
tb = "/mnt/d/06_project/troubleshooter/troubleshooter/tests/diff_handler/tb"
# 可以通过如下接口获取名称映射列表，对npy文件名称映射进行调整
name_list = ts.migrator.get_filename_map_list(ta, tb)
# 通过自定定义一个函数进行list的修改，例如：custom_fun(name_list)
name_list = custom_fun(name_list)
# 将调整后的名称传入比较接口
ts.migrator.compare_npy_dir(name_map_list=name_list)
```

name_list为经过自动映射的列表，类型为list[tuple(str, str)]，例如[(a.npy, a.npy), (b.npy, b.npy),....]。

可以通过调整name_list进行自定义映射和比较。

网络正向结果自动比对

比较MindSpore和PyTorch网络输出是否一致

从PyTorch在迁移到MindSpore过程中，可能存在网络输出精度不一致的情况。需要比较MindSpore和PyTorch网络输出结果是否一致。

此功能实现控制随机性(结果可复现)，使用相同数据或自动生成随机数据，多轮对比MindSpore和PyTorch的输出结果。

网络正向结果自动比对

比较MindSpore和PyTorch网络输出是否

接口参数

参数	类型	说明
ms_net	`mindspore.nn.Cell`	mindspore模型实例
pt_net	`torch.nn.Module`	torch模型实例
input_data	单输入： `Union(tuple[torch.tensor], tuple[mindspore.Tensor], tuple[numpy.ndarray], tuple[str])`；多输入： `list[Union(tuple[torch.tensor], tuple[mindspore.Tensor], tuple[numpy.ndarray], tuple[str])]`	模型输入。模型输入支持 `torch.Tensor`，`mindspore.Tensor`，`np.ndarray` 以及 `str`，每个 `tuple` 中包含一个模型输入；当用户想要同时验证多组数据时，请使用一个列表存放所有输入。
auto_input_data	单输入： `tuple[tuple[numpy.shape, numpy.dtype]]`；多输入： `{'input': tuple[tuple[numpy.shape, numpy.dtype]], 'num':int}`	默认为 `None`，为了方便用户快速验证。用户可以不输入 `input_data`，而是输入 `auto_input_data`，`auto_input_data` 每一个元素为模型输入的 `shape`，如果需要使用多次测试，可以传入一个字典，字典的键为 `'input'` 和 `'num'`，分别表示每次的输入以及输入个数

网络正向结果自动比对

比较MindSpore和PyTorch网络输出是否一致

可以参考troubleshooter/tests/diff_handler/test_netdifffinder.py中的使用方法，或者下面的使用方法：

```
pt_net = ConstTorch()
ms_net = ConstMS()
diff_finder = ts.migrator.NetDifferenceFinder(
    pt_net=pt_net,
    ms_net=ms_net,
    auto_input=(((1, 12), np.float32), ))
diff_finder.compare()
```

输出结果：

```
Saved MindSpore output data at: /Volumes/Samsung_T5/实习/mindspore实习/网络输出结果进行精度比对/toolkits/data/output/MindSpore/result.npy
Saved PyTorch output data at: /Volumes/Samsung_T5/实习/mindspore实习/网络输出结果进行精度比对/toolkits/data/output/PyTorch/result.npy
+-----------------------------------------------------------------------------------------------------------+
|                                        The comparison results of Net                                      |
+--------------+--------------+-------------------------+-------------+-------------------+-----------------------------------+
|   pt data    |   ms data    | results of comparison | match ratio | cosine similarity |          (mean, max, min)         |
+--------------+--------------+-------------------------+-------------+-------------------+-----------------------------------+
| test0-result | test0-result |         False           |     0.00    |      0.99600      | ['0.400000', '0.400000', '0.400000'] |
+--------------+--------------+-------------------------+-------------+-------------------+-----------------------------------+
```

故障分析 (proposer)

- 在线异常建议
- 离线报错分析
- 简洁调用栈

自动生成报错处理建议（在线分析）

结果展示：

```
                                MindSpore FAR(Failure Analysis Report)
+-----------------+-------------------------------------------------------------------------------------------+
|      项目       | 描述                                                                                      |
+-----------------+-------------------------------------------------------------------------------------------+
|   版本信息:     | r1.9                                                                                      |
|   执行模式:     | Graph Mode                                                                                |
|   配置设备:     | CPU                                                                                       |
|   可能原因:     | 静态图的控制流(例如:if...else...)语法要求不同分支返回值的shape相同，不相同时会报错。     |
| 示例错误代码:   | +-----------------------------------------+                                               |
|                 | > def construct(self, x, a, b):                                                          |
|                 | >     if a > b:                                                                           |
|                 | >         return self.relu(x)                                                             |
|                 | >     else:                                                                               |
|                 | >         # shape: (*)->()                                                                |
|                 | >         return self.reducesum(x)                                                        |
|                 | >                ^~~~~~~返回值shape与if分支不一致                                           |
|                 | +-----------------------------------------+                                               |
|   处理建议:     | 检查不同分支的返回结果的shape，如果不相同，请修改至相同shape                              |
|   相关案例:     | 1.编译时报错Shape Join Failed案例:                                                        |
|                 | https://www.mindspore.cn/docs/zh-CN/r1.9/faq/network_compilation.html                     |
|                 | 2.静态图语法:                                                                             |
|                 | https://www.mindspore.cn/docs/zh-CN/r1.9/note/static_graph_syntax_support.html            |
|      ID:        | compiler_id_9                                                                             |
+-----------------+-------------------------------------------------------------------------------------------+
```

等

自动生成报错处理建议（在线分析）

使用方法1 使用装饰器@ts.proposal()

```
import numpy as np
import mindspore
from mindspore import ops, Tensor, nn
import troubleshooter as ts


class Net(nn.Cell):
    def __init__(self):
        super().__init__()
        self.relu = ops.ReLU()
        self.reducesum = ops.ReduceSum()

    def construct(self, x, a, b):
        if a > b:
            return self.relu(x)  # shape: (2, 3, 4, 5), dtype:Float32
        else:
            return self.reducesum(x)  # shape:(), dype: Float32
```

```
# 通过装饰器，装饰在执行函数前
@ts.proposal()
def main():
    input_x = Tensor(np.random.rand(2, 3, 4, 5).astype(np.float32))
    input_a = Tensor(2, mindspore.float32)
    input_b = Tensor(6, mindspore.float32)
    net = Net()
    out = net(input_x, input_a, input_b)
```

自动生成报错处理建议（在线分析）

使用方法2 使用with ts.proposal()

```
import numpy as np
import mindspore
from mindspore import ops, Tensor, nn
import troubleshooter as ts


class Net(nn.Cell):
    def __init__(self):
        super().__init__()
        self.relu = ops.ReLU()
        self.reducesum = ops.ReduceSum()

    def construct(self, x, a, b):
        if a > b:
            return self.relu(x)  # shape: (2, 3, 4, 5), dtype:Float32
        else:
            return self.reducesum(x)  # shape:(), dype: Float32
```

```
# 通过with 来实现片段代码分析
with ts.proposal():
    input_x = Tensor(np.random.rand(2, 3, 4, 5).astype(np.float32))
    input_a = Tensor(2, mindspore.float32)
    input_b = Tensor(6, mindspore.float32)
    net = Net()
    out = net(input_x, input_a, input_b)
```

已生成的报错自动分析（离线分析）

将MindSpore的报错信息，从第一个Traceback (most recent call last): 到最后一行打印的信息，全部拷贝到字符串中，使用raise抛出，即可被proposal捕获并分析。

```
import troubleshooter as ts

@ts.proposal()
def main():
    error = """
    Traceback (most recent call last):
      File "/mnt/d/06_project/trouble-shooter/examples/proposal_demo_1.py", line 25, in <module>
        out = net(input_x, input_a, input_b)
      File "/root/envs/lib/python3.7/site-packages/mindspore/nn/cell.py", line 596, in __call__
        out = self.compile_and_run(*args)
      File "/root/envs/lib/python3.7/site-packages/mindspore/nn/cell.py", line 985, in compile_and_run
        self.compile(*inputs)
      File "/root/envs/lib/python3.7/site-packages/mindspore/nn/cell.py", line 957, in compile
        jit_config_dict=self._jit_config_dict)
      File "/root/envs/lib/python3.7/site-packages/mindspore/common/api.py", line 1131, in compile
        result = self._graph_executor.compile(obj, args_list, phase, self._use_vm_mode())
    ValueError: Cannot join the return values of different branches, perhaps you need to make them equal.
    Shape Join Failed: shape1 = (2, 3, 4, 5), shape2 = ().
    For more details, please refer to https://www.mindspore.cn/search?inputValue=Shape%20Join%20Failed
    ......
    ----------------------------------------------------
    - C++ Call Stack: (For framework developers)
    ----------------------------------------------------
    mindspore/ccsrc/pipeline/jit/static_analysis/static_analysis.cc:850 ProcessEvalResults
    """
    raise ValueError(error)
```

显示简洁异常调用栈(删除部分框架栈信息)

部分框架内部栈信息会影响用户阅读调用栈，此功能以黑名单方式过滤掉部分栈信息，使网络调用栈打

```
Traceback (most recent call last):
  File "/mnt/d/06_project/trouble-shooter/troubleshooter/proposer/proposal_action.py", line 45, in proposal_wrapper
    return func(*args, **kw)
----------------------------------------------------------------------------------------------------
-  [TroubleShooter-Clear Stack] Python Traceback (most recent call last):
----------------------------------------------------------------------------------------------------
Traceback (most recent call last):
  File "/mnt/d/06_project/trouble-shooter/tests/bak/tracker/bak/test_lenet.py", line 52, in test_lent
    out = net(x)
  File "/mnt/d/06_project/trouble-shooter/tests/bak/tracker/bak/test_lenet.py", line 39, in construct
    x = self.fc1(x)
  File "/root/miniconda3/envs/miaoym/lib/python3.7/site-packages/mindspore/nn/layer/basic.py", line 555, in construct
    x = self.matmul(x, self.weight)
  File "/root/miniconda3/envs/miaoym/lib/python3.7/site-packages/mindspore/ops/primitive.py", line 794, in _run_op
    output = _pynative_executor.real_run_op(obj, op_name, args)
  File "/root/miniconda3/envs/miaoym/lib/python3.7/site-packages/mindspore/ops/operations/math_ops.py", line 1473, in check_shape
    raise ValueError(f"For '{cls_name}', the input dimensions must be equal, but got 'x1_col': {x1_col} "
ValueError: For 'MatMul', the input dimensions must be equal, but got 'x1_col': 144 and 'x2_row': 400. And 'x' shape [1, 144](transp
```

```
  File "/root/miniconda3/envs/miaoym/lib/python3.7/site-packages/mindspore/ops/primitive.py", line 494, in __check__
    fn(*(x[track] for x in args))
  File "/root/miniconda3/envs/miaoym/lib/python3.7/site-packages/mindspore/ops/operations/math_ops.py", line 1473, in check_shape
    raise ValueError(f"For '{cls_name}', the input dimensions must be equal, but got 'x1_col': {x1_col} "
ValueError: For 'MatMul', the input dimensions must be equal, but got 'x1_col': 144 and 'x2_row': 400. And 'x' shape [1, 144](transp
```

开启后

显示简洁异常调用栈(删除部分框架栈信息)

如何使用

配置*print_clear_stack=True*参数（默认*False*)

```
@ts.proposal(print_clear_stack=True)
def test_lent():
    context.set_context(mode=context.PYNATIVE_MODE, device_target="CPU")
    x = np.arange(1 * 24 * 24).reshape(1, 1, 24, 24)
    x = Tensor(x, ms.float32)
    net = LeNet5()
    out = net(x)
```

跟踪调试(tracker)

- 网络运行信息
- 获取INF/NAN值抛出点
- 黑白名单过滤追踪信息
- 跟踪API报错详细执行过程

跟踪调试

在网络问题定位中，例如算子shape报错/结果出现nan值等场景时，通常需要进行pdb调试、查看调用栈等手段。

troubleshooter提供了跟踪网络执行的tracking功能，可以跟踪实际执行过程，打印丰富的执行信息。（当前仅对pynative模式和图模式下的Python阶段有效）

网络运行信息(结构、shape)

效果

```
============== Starting Training ==============
Source path:... /mnt/d/06_project/trouble-shooter/examples/lenet/src/lenet.py
Starting var:.. self = LeNet5<  (conv1): Conv2d<input_channels=1, output_channels=6, kernel_size...ue>  (fc3): Dense<input_channels=84, output_channels=10, has_bias=True>  >
Starting var:.. x = Tensor(shape=[32, 32, 32, 1], dtype=Float32, value=[[[[-4.24212933e-01], ...],   ...   [-4.24212933e-01],   [-4.24212933e-01],   [-4.24212933e-01]]]])
19:51:02.074695 call        50     def construct(self, x):
19:51:02.075004 line        51         x = self.transpose(x, (0, 3, 1, 2))
Modified var:.. x = Tensor(shape=[32, 1, 32, 32], dtype=Float32, value=[[[[-4.24212933e-01, -...-4.24212933e-01 ... -4.24212933e-01, -4.24212933e-01, -4.24212933e-01]]]])
19:51:02.192380 line        52         x = self.conv1(x)
Modified var:.. x = Tensor(shape=[32, 6, 28, 28], dtype=Float32, value=[[[[ 1.79276634e-02,  ...-1.63824446e-02 ... -1.63824446e-02, -1.63824446e-02, -1.63824446e-02]]]])
19:51:02.230425 line        53         x = self.relu(x)
Modified var:.. x = Tensor(shape=[32, 6, 28, 28], dtype=Float32, value=[[[[ 1.79276634e-02,  ... 0.00000000e+00 ...  0.00000000e+00,  0.00000000e+00,  0.00000000e+00]]]])
19:51:02.273478 line        54         x = self.max_pool2d(x)
Modified var:.. x = Tensor(shape=[32, 6, 14, 14], dtype=Float32, value=[[[[ 1.79276634e-02,  ... 0.00000000e+00 ...  0.00000000e+00,  0.00000000e+00,  0.00000000e+00]]]])
19:51:02.286534 line        55         x = self.conv2(x)
Modified var:.. x = Tensor(shape=[32, 16, 10, 10], dtype=Float32, value=[[[[-8.87135230e-03, ... 5.21629537e-03 ...  8.06936715e-03,  7.11881509e-03,  4.42999927e-03]]]])
19:51:02.299206 line        56         x = self.relu(x)
Modified var:.. x = Tensor(shape=[32, 16, 10, 10], dtype=Float32, value=[[[[ 0.00000000e+00, ... 5.21629537e-03 ...  8.06936715e-03,  7.11881509e-03,  4.42999927e-03]]]])
19:51:02.303392 line        57         x = self.max_pool2d(x)
Modified var:.. x = Tensor(shape=[32, 16, 5, 5], dtype=Float32, value=[[[[ 0.00000000e+00,  0...3,  5.21629537e-03,  5.83564164e-03,  8.06936715e-03,  9.98822320e-03]]]])
19:51:02.305934 line        58         if not self.include_top:
19:51:02.306866 line        60         x = self.flatten(x)
Modified var:.. x = Tensor(shape=[32, 400], dtype=Float32, value=[[ 0.00000000e+00,  0.000000...,  0.00000000e+00 ...  5.83564164e-03,  8.06936715e-03,  9.98822320e-03]])
19:51:02.382671 line        61         x = self.relu(self.fc1(x))
Modified var:.. x = Tensor(shape=[32, 120], dtype=Float32, value=[[ 0.00000000e+00,  7.809398...,  1.54157146e-03 ...  1.00187608e-03,  0.00000000e+00,  5.11248969e-03]])
19:51:02.411025 line        62         x = self.relu(self.fc2(x))
Modified var:.. x = Tensor(shape=[32, 84], dtype=Float32, value=[[ 0.00000000e+00,  0.0000000...,  0.00000000e+00 ...  0.00000000e+00,  0.00000000e+00,  0.00000000e+00]])
19:51:02.427581 line        63         x = self.fc3(x)
Modified var:.. x = Tensor(shape=[32, 10], dtype=Float32, value=[[-4.44369243e-06, -3.7606334...,  8.77573075e-06 ...  7.36864022e-05, -4.64724144e-05, -7.39026073e-05]])
19:51:02.435671 line        64         return x
19:51:02.436213 return      64         return x
Return value:.. Tensor(shape=[32, 10], dtype=Float32, value=[[-4.44369243e-06, -3.7606334...,  8.77573075e-06 ...  7.36864022e-05, -4.64724144e-05, -7.39026073e-05]])
Elapsed time: 00:00:00.363648
```

网络运行信息(结构、shape)

使用方式：

```python
import mindspore.nn as nn
from mindspore import ops
from mindspore.common.initializer import Normal
import troubleshooter as ts

class LeNet5(nn.Cell):
    def __init__(self, num_class=10, num_channel=1, include_top=True):
        super(LeNet5, self).__init__()
        self.conv1 = nn.Conv2d(num_channel, 6, 5, pad_mode='valid')
        self.conv2 = nn.Conv2d(6, 16, 5, pad_mode='valid')
        self.relu = nn.ReLU()
        self.max_pool2d = nn.MaxPool2d(kernel_size=2, stride=2)
        self.include_top = include_top
        self.transpose = ops.Transpose()
        if self.include_top:
            self.flatten = nn.Flatten()
            self.fc1 = nn.Dense(16 * 5 * 5, 120, weight_init=Normal(0.02))
            self.fc2 = nn.Dense(120, 84, weight_init=Normal(0.02))
            self.fc3 = nn.Dense(84, num_class, weight_init=Normal(0.02))
```

1. 使用装饰器 @ts.tracking()

```python
# 跟踪construct函数执行
@ts.tracking()
def construct(self, x):
    x = self.transpose(x, (0, 3, 1, 2))
    x = self.conv1(x)
    x = self.relu(x)
    x = self.max_pool2d(x)
    x = self.conv2(x)
    x = self.relu(x)
    x = self.max_pool2d(x)
    if not self.include_top:
        return x
    x = self.flatten(x)
    x = self.relu(self.fc1(x))
    x = self.relu(self.fc2(x))
    x = self.fc3(x)
    return x
```

2. 使用with ts.tracking()

```python
def construct(self, x):
    x = self.transpose(x, (0, 3, 1, 2))
    x = self.conv1(x)
    x = self.relu(x)
    x = self.max_pool2d(x)
    x = self.conv2(x)
    x = self.relu(x)
    # 使用with 仅打印部分输出
    with ts.tracking():
        x = self.max_pool2d(x)
        if not self.include_top:
            return x
        x = self.flatten(x)
        x = self.relu(self.fc1(x))
        x = self.relu(self.fc2(x))
        x = self.fc3(x)
    return x
```

获取INF/NAN值抛出点

效果展示

x = self.sqrt(y) **出现** *nan*, **给出** *“User Warning 'nan' is detected”***报错。**

```
Source path:... /mnt/d/06_project/trouble-shooter/examples/tracking_demo_1.py
20:06:10.780427 call        20 def nan_func():
    Starting var:.. self = Net<>
    Starting var:.. input_x = Tensor(shape=[2, 2], dtype=Float64, value=[[ 0.00000000e+00, -1.00000000e+00], [ 4.00000000e+00,  3.00000000e+00]])
    20:06:10.786076 call        14     def construct(self, input_x):
    20:06:10.786147 line        15         y = self.matmul(input_x, input_x)
    New var:..... y = Tensor(shape=[2, 2], dtype=Float64, value=[[-4.00000000e+00, -3.00000000e+00], [ 1.20000000e+01,  5.00000000e+00]])
    20:06:12.815509 line        16         x = self.sqrt(y)
2022-12-05 20:06:13,372 - troubleshooter.log - WARNING - User Warning 'NAN' is detected, Key is x, Value is Tensor(shape=[2, 2], dtype=Float64, value=[[          -nan,
       Elapsed time: 00:00:02.593829

进程已结束,退出代码0
```

获取INF/NAN值抛出点

使用方法：

使用ts.tracking的“check_keyword”参数

- check_keyword：用于设置需要检查的值
- check_mode：用于设置检查模式,1:检查到值即退出并打印用户告警日志（默认）,2:检查到值不退出，仅打印用户告警日志

```
import mindspore
import numpy as np
from mindspore import nn, ops, Tensor
import troubleshooter as ts
mindspore.set_context(mode=mindspore.PYNATIVE_MODE)
class Net(nn.Cell):
    def __init__(self):
        super().__init__()
        self.sqrt = ops.Sqrt()
        self.matmul = ops.MatMul()

    def construct(self, input_x):
        y = self.matmul(input_x, input_x)
        x = self.sqrt(y)
        return x
```

```
# 配置检查值
@ts.tracking(check_mode=1, check_keyword='nan')
def nan_func():
    input_x = Tensor(np.array([[0.0, -1.0], [4.0, 3.0]]))
    k = 3.0
    net = Net()
    print(net(input_x))
nan_func()
```

按照路径黑白名单过滤跟踪信息

结果展示：

配置路径白名单,仅跟踪'layer/conv.py'文件中的代码

```
Source path:... /mnt/d/06_project/trouble-shooter/examples/tracking_demo_2.py
20:23:45.188025 call         10 def main():
    Source path:... /root/miniconda3/envs/miaoym/lib/python3.7/site-packages/mindspore/nn/layer/conv.py
    Starting var:.. self = REPR FAILED
    Starting var:.. in_channels = 3
    Starting var:.. out_channels = 2
    Starting var:.. kernel_size = 3
    Starting var:.. stride = 1
    Starting var:.. pad_mode = 'valid'
```

按照路径黑白名单过滤跟踪信息

方法1：使用ts.tracking的"path_wl"白名单参数

```
import mindspore
import numpy as np
from mindspore import nn, ops, Tensor
import troubleshooter as ts

mindspore.set_context(mode=mindspore.PYNATIVE_MODE)

# 使用path_wl白名单，仅打印白名单文件中的代码跟踪
@ts.tracking(level=1, path_wl=['layer/conv.py'])
def main():
    conv = nn.Conv2d(3, 2, 3, pad_mode='valid', weight_init="ones")
    relu = nn.ReLU()
    seq = nn.SequentialCell([conv, relu])
    x = Tensor(np.ones([1, 3, 4, 4]), dtype=mindspore.float32)
```

按照路径黑白名单过滤跟踪信息

方法2：使用ts.tracking的“path_bl”黑名单参数

```
# 黑名单与level 可同时使用，先通过level过滤，再通过黑名单过滤
@ts.tracking(level=2, color=False, path_bl=['layer/activation.py'])
def main():
    context.set_context(mode=context.PYNATIVE_MODE)
    conv = nn.Conv2d(3, 2, 3, pad_mode='valid', weight_init="ones")
    relu = nn.ReLU()
    seq = nn.SequentialCell([conv, relu])
    x = Tensor(np.ones([1, 3, 4, 4]), dtype=mindspore.float32)
    output = seq(x)
    print(output)
```

跟踪API报错的详细执行过程

当API报错时，我们仅能看到有限的堆栈信息，有时需要了解API的调用流程和参数传递&变化过程，以定位报错的原因，此场景下，可以应用tracking功能进行错误跟踪。

跟踪可以通过level参数控制跟踪信息
- level=1(默认) 仅跟踪用户脚本中的construct函数
- level=2 跟踪用户脚本以及MindSpore主要的Python执行代码
- level=3 跟踪用户脚本以及MindSpore的全部Python执行代码
- level=4 跟踪用户脚本、MindSpore、Python库以及第三方包的所有Python执行代码

跟踪API报错的详细执行过程

当API报错时，我们仅能看到有限的堆栈信息，有时需要了解API的调用流程和参数传递&变化过程，以定位报错的原因，此场景下，可以应用tracking功能进行错误跟踪。

跟踪可以通过level参数控制跟踪信息
- level=1(默认) 仅跟踪用户脚本中的construct函数
- level=2 跟踪用户脚本以及MindSpore主要的Python执行代码
- level=3 跟踪用户脚本以及MindSpore的全部Python执行代码
- level=4 跟踪用户脚本、MindSpore、Python库以及第三方包的所有Python执行代码

跟踪API报错的详细执行过程

案例

3）结合代码和跟踪信息可知，self.weight是根据dense的参数in_channels和out_channels决定的，定位到问题原因。

```
18:11:51.547729 line        46              fn(self, *args, **kwargs)
    Source path:... /root/miniconda3/envs/miaoym/lib/python3.7/site-packages/mindspore/nn/layer/basic.py
    Starting var:.. self = REPR FAILED
    Starting var:.. in_channels = 5
    Starting var:.. out_channels = 4
    Starting var:.. weight_init = 'normal'
    Starting var:.. bias_init = 'zeros'
    Starting var:.. has_bias = True
    Starting var:.. activation = None
    Starting var:.. __class__ = <class 'mindspore.nn.layer.basic.Dense'>
```

```
    Exception:..... ValueError: For 'MatMul', the input dimensions must be equal, but got 'x1...d 'x' shape [2, 3](transpose_a=False), 'y' shape [4, 5](transpose_b=True).
    Call ended by exception
Source path:... /mnt/d/06_project/trouble-shooter/examples/rank_0/tracking_demo_3.py
18:06:34.035188 exception   15          return self.fc(x)
Exception:..... ValueError: For 'MatMul', the input dimensions must be equal, but got 'x1...d 'x' shape [2, 3](transpose_a=False), 'y' shape [4, 5](transpose_b=True).
```

Next:

反向精度定位

API级别精度自动比对

静态图API级别数据保存

…

欢迎加入我们，共同提高MindSpore易用性！

Gitee:

https://gitee.com/MindSpore/toolkits/tree/master/troubleshooter

Thank you!