The results are not reproducible. (codes are not complete ?) #3

New issue

Open **TianweiXing** opened this issue on Jul 28 · 1 comment

TianweiXing commented on Jul 28

Hi there,
Is this the latest repo for Deep Compressive Offloading? Trying to reproduce the result claimed in the paper but seems the codes are missing.
Do you have any plan to finish the repo?

yoshitomo-matsubara commented 20 seconds ago

If you're a reviewer/chair and about to ask authors of papers in review for comparison to this work, please consider comments here.

Though this work is awarded best paper award at SenSys 2021, I agree with **@TianweiXing** that the results reported in the paper are not reproducible with the current version of this repository.

The main reasons are as follows:

1. This repository seems to have only one configuration (1 bottleneck point while showing 5-6 points) for ImageNet dataset (no speech recognition)
2. This repository lacks of instructions to install dependencies (required packages and their versions) and commands to run their code.
3. There are no trained models provided to confirm the accuracy without training. While many studies report Top-1 accuracy, we can only see Top-5 accuracy (at least one of the top 5 guesses is correct) numbers in the paper.
4. It looks not clear how they measured the data size of bottleneck reported in kilobyte.

For instance, The following minimal code shows reimplementation of the "Residual Transposed Convolution" module based on Figure 3 in their paper, which will fail at the add operation (+) in the figure since the tensor size at the top forward path doesn't match that from the bottom forward path in the figure.

```
import torch
from torch import nn

# Here use some random number of channels since number of channels is not described in the
paper
ch = 3
z = torch.rand(2, ch, 100, 100)

# "Residual Transposed Convolution" in Figure 3
top_deconv_k3x3_s2x2 = nn.ConvTranspose2d(ch, ch, (3, 3), (2, 2))
top_deconv_k3x3_s1x1 = nn.ConvTranspose2d(ch, ch, (3, 3), (1, 1))
bottom_deconv_k3x3_s1x1 = nn.ConvTranspose2d(ch, ch, (3, 3), (1, 1))


# Top forward path
top_z1 = top_deconv_k3x3_s2x2(z)
print(top_z1.shape)
# torch.Size([2, 3, 201, 201])
top_z2 = top_deconv_k3x3_s1x1(top_z1)
print(top_z2.shape)
# torch.Size([2, 3, 203, 203])

# Bottom forward path
bottom_z = bottom_deconv_k3x3_s1x1(z)
print(bottom_z.shape)
# torch.Size([2, 3, 102, 102])

# top_z2.shape doesn't match bottom_z.shape, thus add operation (+) in the figure cannot
work for top_z2 to bottom_z
out_residual_trans_conv = top_z2 + bottom_z

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
RuntimeError: The size of tensor a (203) must match the size of tensor b (102) at non-
singleton dimension 3
```

Dear @yscacaca ,

Top-1 accuracy except Figure 1 (b), and either the numbers or bottleneck point used in Figure 1 (b) is not clear.
Also, people cannot reproduce the tradeoff between accuracy and data size with the current version of this repository.

It would be really appreciated if you can address the critical issues so that people can compare their methods to this DeepCOD study.

Best regards

None yet

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

2 participants