trtexec dynamic batch size #976

Closed **zirui** opened this issue on Dec 20, 2020 · 26 comments

Labels Component: ONNX Release: 7.x Topic: Dynamic Shape triaged

zirui commented on Dec 20, 2020 • edited

Description

I tried to convert my onnx model to tensorRT model with trtexec , and i want the batch size to be dynamic,
but failed with two problems:

1. trtrexec with `maxBatch` param failed
2. tensorRT model was converted successfully after specify shape params(minShapes/maxShapes/optShapes), but error occurs when load model back with trtexec

Environment

TensorRT Version: 7.2.1
GPU Type: Telsa P40
Nvidia Driver Version: 418.40.04
CUDA Version: cuda11.1
CUDNN Version: 8.04
Operating System + Version: 18.04.5 LTS
Python Version (if applicable): 3.6.6
Baremetal or Container (if container which image + tag): nvcr.io/nvidia/tensorrt:20.11-py3

Relevant Files

Steps To Reproduce

1. Convert onnx model with `maxBatch`

```
trtexec --onnx=bert.onnx --maxBatch 16 --saveEngine=bert.trt
```

error logs returned by trtexec

```
[12/20/2020-07:31:45] [E] stoi
```

2.

(1) Convert to tensorRT model by specify Shape params

```
trtexec --onnx=bert.onnx \
        --explicitBatch \
        --saveEngine=bert_model.trt \
        --minShapes=input_ids:1x128,attention_mask:1x128,token_type_ids:1x128 \
        --optShapes=input_ids:8x128,attention_mask:8x128,token_type_ids:8x128 \
        --maxShapes=input_ids:16x128,attention_mask:16x128,token_type_ids:16x128 \
```

(2) Load model with trtexec

```
trtexec --loadEngine=bert_model.trt
```

```
[12/20/2020-07:19:07] [E] Error opening engine file:bert_model.trt
[12/20/2020-07:19:07] [E] Engine creation failed
[12/20/2020-07:19:07] [E] Engine set up failed
&&&& FAILED TensorRT.trtexec # trtexec --loadEngine=bert_model.trt
```

nvpohanh commented on Dec 21, 2020

(1) `maxBatch` cannot be used with ONNX, which only supports explicit batch dimension. So we'll focus on (2).

(2) When you load the engine, you need to pass in the actual shape you would like to run the inference with by adding `--shapes=input_ids:<N>x128,attention_mask:<N>x128,token_type_ids:<N>x128` , where `<N>` is the actual batch size for inference. Could you try this and report if it works? Thanks

👍 1

zirui commented on Dec 21, 2020 • edited

thanks **@nvpohanh**
i added '--shapes' as you said, and trtexec run inference successfully.

```
trtexec --loadModel=bert_model.trt \
        --shapes=input_ids:32x128,attention_mask:32x128,token_type_ids:32x128
```

but the inference time is more than 50x than trt model with fixed `batch size 1` (converted without specify minShape/Maxshape parms),
so now i have another two questions:

1. does the inference time returned by trtexec is the `total time of inference <N> samples` rather than the time of inference every `single sample` ?
2. is it normal that dynamic batch model(N >1) is slower than model with fixed batch size of 1 when inference single sample

GPU latency:

model	batch size	inference time (ms)
default model(build with bs=1)	1	7
dynamic batch model(build with bs=64)	1	20
dynamic batch mode(build with bs=64)l	32	380

message of dynamic batch model when run inference with shapes params

```
[12/21/2020-13:12:28] [I] Average on 10 runs - GPU latency: 380.035 ms - Host latency:
380.084 ms (end to end 759.362 ms, enqueue 236.376 ms)
[12/21/2020-13:12:28] [I] Host Latency
[12/21/2020-13:12:28] [I] min: 379.533 ms (end to end 754.411 ms)
[12/21/2020-13:12:28] [I] max: 380.833 ms (end to end 761.23 ms)
[12/21/2020-13:12:28] [I] mean: 380.084 ms (end to end 759.362 ms)
[12/21/2020-13:12:28] [I] median: 379.969 ms (end to end 759.445 ms)
[12/21/2020-13:12:28] [I] percentile: 380.833 ms at 99% (end to end 761.23 ms at 99%)
[12/21/2020-13:12:28] [I] throughput: 0 qps
[12/21/2020-13:12:28] [I] walltime: 4.17524 s
[12/21/2020-13:12:28] [I] Enqueue Time
[12/21/2020-13:12:28] [I] min: 231.493 ms
[12/21/2020-13:12:28] [I] max: 237.601 ms
[12/21/2020-13:12:28] [I] median: 236.726 ms
[12/21/2020-13:12:28] [I] GPU Compute
[12/21/2020-13:12:28] [I] min: 379.502 ms
[12/21/2020-13:12:28] [I] max: 380.762 ms
[12/21/2020-13:12:28] [I] mean: 380.035 ms
[12/21/2020-13:12:28] [I] median: 379.915 ms
[12/21/2020-13:12:28] [I] percentile: 380.762 ms at 99%
[12/21/2020-13:12:28] [I] total compute time: 3.80035 s
```

nvpohanh commented on Dec 22, 2020

trtexec returns the runtime per inference, where an "inference" is a query of batch_size=N which you specified.

> is it normal that dynamic batch model(N >1) is slower than model with fixed batch size of 1 when inference single sample

It is possible, although in this case I would be surprised by this difference. Could you share what commands you use to build the "default model" and the "dynamic batch (batch=1) model"? The default should be identical (or equivalent to) using `--minShapes=input_ids:1x128,attention_mask:1x128,token_type_ids:1x128 --optShapes=input_ids:1x128,attention_mask:1x128,token_type_ids:1x128 --maxShapes=input_ids:1x128,attention_mask:1x128,token_type_ids:1x128` flags.

zirui commented on Dec 22, 2020 • edited

Maybe I am not describe clearly, `dynamic batch (batch=1) model` means model `build` with `maxbatch size 64` , and `inference` with `batch size 1`
i used below commands to build and test models:

build model
default model:

```
trtexec --onnx=bert_model.onnx --saveEngine=model_default.trt
```

dynamic batch model:

```
trtexec --onnx=bert_model.onnx \
        --explicitBatch \
        --saveEngine=batch_model.trt \
        --minShapes=input_ids:1x128,attention_mask:1x128,token_type_ids:1x128 \
        --optShapes=input_ids:8x128,attention_mask:8x128,token_type_ids:8x128 \
        --maxShapes=input_ids:64x128,attention_mask:64x128,token_type_ids:64x128 \
```

inference test
default model:

```
trtexec --loadEngine=model_default.trt
```

batch model(inference with batch size of 1 and 32)

```
trtexec --loadEngine=batch_model.trt --
shapes=input_ids:1x128,attention_mask:1x128,token_type_ids:1x128
trtexec --loadEngine=batch_model.trt --
shapes=input_ids:32x128,attention_mask:32x128,token_type_ids:32x128
```

nvpohanh commented on Dec 22, 2020

I see. Since "optShapes" is batch_size=8, TRT will use batch_size=8 to choose the fastest tactics, which may not be optimal for batch_size=1 and batch_size=32. Could you try to change optShapes from 8 to 1 or 32 and see if that helps improve the inference time?

zirui commented on Dec 22, 2020

By changed optShapes from 8 to 1 and 32, I build another two models ,
and got the preformance report below:

model	build-maxShapes	build-optShapes	inference-batch_size	median GPU latency (ms)
default model	default(1)	default(1)	1	7
dynamic batch model	64	8	1	20
dynamic batch mode	64	8	32	380
dynamic batch mode	64	1	1	20.1
dynamic batch mode	64	1	32	379.4
dynamic batch mode	64	32	1	20
dynamic batch mode	64	32	32	380.3

it seems like that the `optShapes` doesn't make any difference

nvpohanh commented on Dec 22, 2020

I see. That probably means there are some faster tactics taking advantage of static shapes but won't work with dynamic shapes. Could you try setting max=opt=min=inference=32 and see if that helps the latency for batch_size=32 case?

zirui commented on Dec 22, 2020 • edited

i tried `max=opt=min=inference=32` , then the gpu latency increase to 472ms!
and i noticed that there was warning message in the process of building model with trtexec

```
[I] [TRT] Some tactics do not have sufficient workspace memory to run. Increasing
workspace size may increase performanc
```

but trtexec always return `[E] stoi` when add the `workspace` param. eg:

```
 trtexec --onnx=bert.onnx --saveEngine=bert.trt --workspace 4096  --
shapes=input_ids:32x128,attention_mask:32x128,token_type_ids:32x128
```

ttyio added **Release: 7.x** Topic: Dynamic Shape triaged Component: ONNX labels on Jan 11

nvpohanh commented on Jan 14

@zirui Could you try `--workspace=4096` instead of `--workspace 4096` ? Thanks

zirui commented on Jan 14 • edited

> **@zirui** Could you try `--workspace=4096` instead of `--workspace 4096` ? Thanks

it works!
i think we need more detail error logs, the message `[E] stoi` is too confusing..

And when test with the new model built with `--workspace=4096` ,
the inference time reduce to 378ms, which seems better than 472 , but no big difference with 380ms(model build with opt= 1/8/32)

nvpohanh commented on Jan 14

I see. I think now the mystery is: does `max=opt=min=inference=1` give you 7ms or 20ms?

zirui commented on Jan 14 • edited

i rebuild models with different `min/opt/max shapes(include max=opt=min=1)` , and update the performance report below:

min-opt-max shapes:

```
1-1-1
1-1-64
1-8-64
1-32-64
32-32-32
```

performance test

model	build/min-opt-max	inference/batch_size	mean GPU latency (ms)
default model	default	1	7.01869
dynamic batch mode	1-1-1	1	19.7453
dynamic batch mode	1-1-1	32	19.745
dynamic batch mode	1-8-64	1	20.0619
dynamic batch mode	1-8-64	32	379.261
dynamic batch mode	1-1-64	1	19.9791
dynamic batch mode	1-1-64	32	381.134
dynamic batch mode	1-32-64	1	20.0532
dynamic batch mode	1-32-64	32	378.898

model	build/min-opt-max	inference/batch_size	mean GPU latency (ms)
dynamic batch mode	32-32-32	1	378.634
dynamic batch mode	32-32-32	32	378.695
dynamic batch mode	8-32-32	1	378.634
dynamic batch mode	8-32-32	32	380.618

so, from the table, we can see that:
model with `max=opt=min=inference=1` cost around 20 ms, while the default model cost 7ms

👍 2

nvpohanh commented on Jan 15

@zirui Would it be possible that you can provide the onnx file so that we can debug this gap? Thanks

zirui commented on Jan 20

> **@zirui** Would it be possible that you can provide the onnx file so that we can debug this gap? Thanks

I am unable to provide the model file due to the restriction of my private network environment,
but i will take time to train a new model in public network environment to reproduce the problem this week

nvpohanh commented on Jan 20

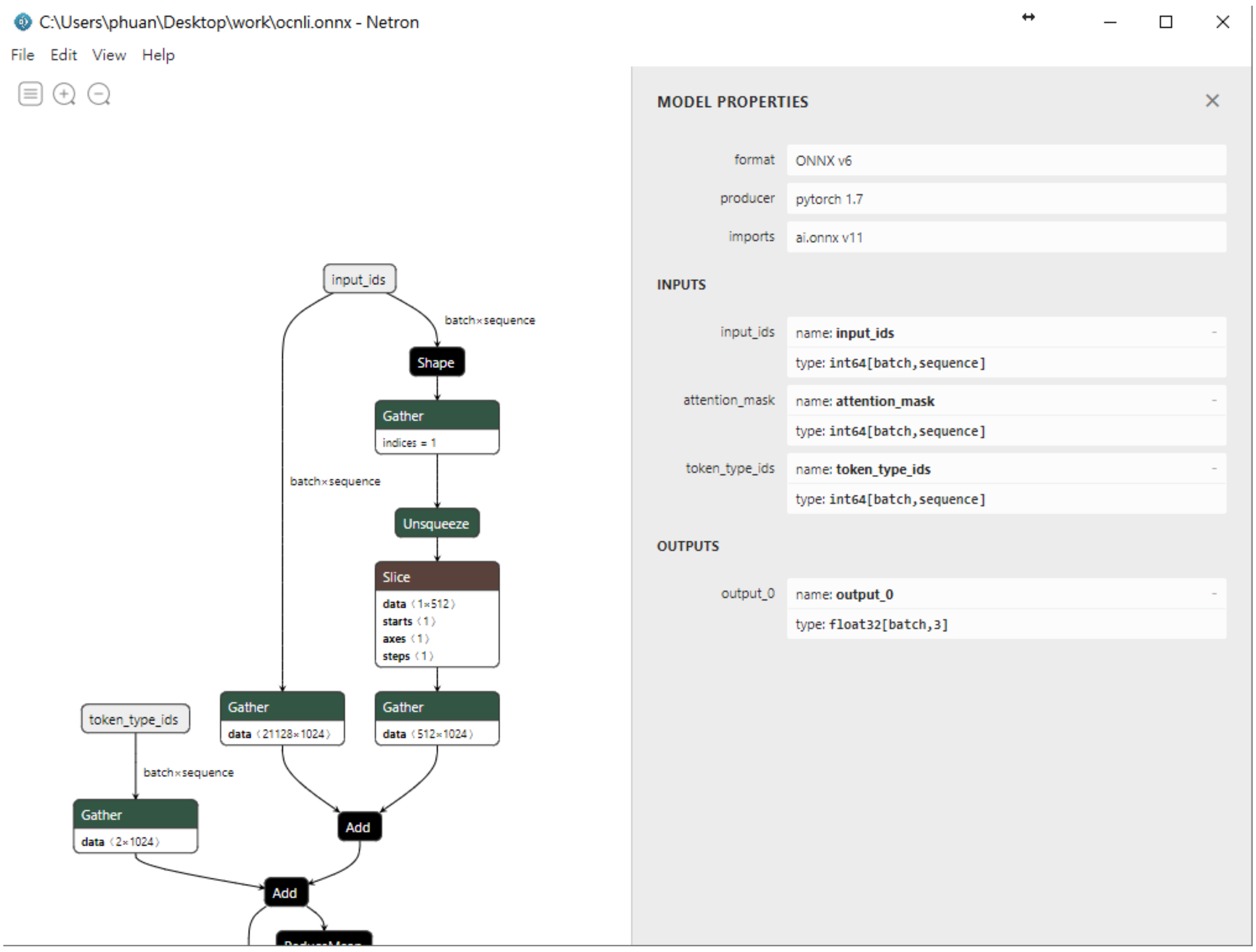
Thanks. I think the main focus would be: why is 1-1-1 different from the default model?

zirui commented on Jan 25

hi, **@nvpohanh**
I have rebuilt the onnx model and uploaded the model to the google drive

nvpohanh commented on Jan 25

@zirui I see that your onnx file input shapes have two dynamic axes: `[batch, sequence]` , so when you don't pass in any shapes, trtexec actually sets the dynamic axes to 1 and throws a warning in the logs. That means, your 7ms measurement was in fact measuring the performance when the input shape is [1, 1].



MODEL PROPERTIES

format	ONNX v6
producer	pytorch 1.7
imports	ai.onnx v11

INPUTS

input_ids	name: **input_ids**
	type: `int64[batch,sequence]`
attention_mask	name: **attention_mask**
	type: `int64[batch,sequence]`
token_type_ids	name: **token_type_ids**
	type: `int64[batch,sequence]`

OUTPUTS

output_0	name: **output_0**
	type: `float32[batch,3]`

zirui commented on Jan 25

I saw the warning logs when build `default` tensorrt model:

```
[W] Dynamic dimensions required for input: token_type_ids, but no shapes were provided.
Automatically overriding shape to: 1x1
```

but i found that the model could run inference by specify `[batch, sequence]` to `1 x 128` without any error or warning,
and the inference time is around 7ms

```
trtexec --loadEngine=default.trt --
shapes=input_ids:1x128,attention_mask:1x128,token_type_ids:1x128
```

```
[01/25/2021-09:28:53] [I] === Inference Options ===
[0/2587]
[01/25/2021-09:28:53] [I] Batch: Explicit
[01/25/2021-09:28:53] [I] Input inference shape: token_type_ids=1x128
[01/25/2021-09:28:53] [I] Input inference shape: input_ids=1x128
[01/25/2021-09:28:53] [I] Input inference shape: attention_mask=1x128
...
[01/25/2021-09:29:23] [I] GPU Compute
[01/25/2021-09:29:23] [I] min: 7.00513 ms
[01/25/2021-09:29:23] [I] max: 7.17107 ms
[01/25/2021-09:29:23] [I] mean: 7.03121 ms
[01/25/2021-09:29:23] [I] median: 7.02771 ms
[01/25/2021-09:29:23] [I] percentile: 7.08594 ms at 99%
```

so does this mean the default model run inference with fixed `1 x 1` input, and the `shapes` params actually dose not work as expected?

nvpohanh commented on Jan 25

This is because the engine has been built with a **fixed** shape of 1x1, so the input shape is no longer dynamic. That also means trtexec won't try to set the input shape again (it's already static, anyway). I do think we should improve trtexec so that it throws errors (or at least warnings) if user-provided shapes are not used.

zirui commented on Feb 1

@nvpohanh
thanks for your kindly reply, i'm closing this issue

zirui closed this on Feb 1

Vampire-Vx commented on Jul 20

> (1) `maxBatch` cannot be used with ONNX, which only supports explicit batch dimension. So we'll focus on (2).
>
> (2) When you load the engine, you need to pass in the actual shape you would like to run the inference with by adding `--shapes=input_ids:<N>x128,attention_mask:<N>x128,token_type_ids:<N>x128`, where `<N>` is the actual batch size for inference. Could you try this and report if it works? Thanks

Hey, you said that "ONNX, which only supports explicit batch dimension."
I try the command `trtexec --onnx=bert_model.onnx --saveEngine=model_default.trt --batch=5`. It works and I can loadEngine to run inference for batch<=5. And I am confused is it implicit or explicit?

nvpohanh commented on Jul 20

Starting from TRT 7 (I think), ONNX is always explicit batch. Using `--batch` has no effect. You can check the throughput or latency numbers and see that it makes no difference

👍 2

KDr2 commented on Jul 30

My input shape is 1x2x2x20, and the output shape is (1,). I can use `trtexec --shape=input:<N>x2x2x20` to specify a batch size N, but the output is still (1,), how to specify its shape to (N,)?

Thanks.

nvpohanh commented on Jul 30

@KDr2 When you export the ONNX file, could you specify the batch dimension to be a dynamic dimension? If you are using PyTorch, then use the `dynamic_axes` argument in `torch.onnx.export()` function call: https://pytorch.org/tutorials/advanced/super_resolution_with_onnxruntime.html

Vampire-Vx mentioned this issue on Jul 30

trtexec Implicit Batch for ONNX model ? #1387

Closed

KDr2 commented on Jul 31

Hi **@nvpohanh**, my onnx model is converted from a tensorflow saved mode:

```
      inputs['input_2'] tensor_info:
          dtype: DT_INT32
          shape: (-1, 2, 2, 20)
          name: texts:0
```

Above is the tensor info of the input.

And then I used `python -m tf2onnx.convert --saved-model saved_model/ --output 2.onnx --opset=11` to convert it to an ONNX. Is there a way to make the batch dimension dynamic in this approach?

Thanks.

nvpohanh commented on Aug 1

I am not very familiar with tf2onnx, but it seems that you can try this:

> --inputs, --outputs
> TensorFlow model's input/output names, which can be found with summarize graph tool. Those names typically end with :0, for example --inputs input0:0,input1:0. Inputs and outputs are not needed for models in saved-model format. Some models specify placeholders with unknown ranks and dims which can not be mapped to onnx. In those cases one can add the shape after the input name inside [], for example --inputs X:0[1,28,28,3]. Use -1 to indicate unknown dimensions.

https://github.com/onnx/tensorflow-onnx#--inputs---outputs

L1aoXingyu mentioned this issue 26 days ago

How to export a TRT model with implicit batch size? (rather than explicit batch size)
JDAI-CV/fast-reid#551
Open

Assignees

No one assigned

Labels

Component: ONNX Release: 7.x Topic: Dynamic Shape triaged

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

5 participants