BEST PRACTICES FOR CUDNN

DG-09678-001_v7.6.5 | November 2019

Best Practices

TABLE OF CONTENTS

Chapter 1. INTRODUCTION

Attention These guidelines are applicable to 3D convolution and deconvolution functions starting in cuDNN v7.6.3.

This document provides guidelines for setting the cuDNN library parameters to enhance the performance of 3D convolutions. Specifically, these guidelines are focused on settings such as filter sizes, padding and dilation settings. Additionally, an application-specific use-case, namely, medical imaging, are presented to demonstrate the performance enhancement of 3D convolutions with these recommended settings.

Specifically, these guidelines are applicable to the following functions and their associated data types:

- cudnnConvolutionForward()
- cudnnConvolutionBackwardData()
- cudnnConvolutionBackwardFilter()

For more information, see the cuDNN Developer Guide and cuDNN API.

Chapter 2. BEST PRACTICES FOR MEDICAL IMAGING

To optimize your performance in your model, ensure you meet the following general guidelines:

Layout

The layout is in NCHW format.

Filter size

The filter size is **`Tx1x1`**, **`Tx2x2`**, **`Tx3x3`**, **`Tx5x5`**, where **`T`** is a positive integer. There are additional limits for the value of **`T`** in **`wgrad`** and strided **`dgrad`**.

Padding

(filter size // 2), for example, 0x0x0 for 1x1x1 filter, 1x1x1 for 3x3x3 filter

Stride

Arbitrary for forward and backward filter; **`dgrad`**/**`deconv`**: 1x1x1 or 2x2x2 with 2x2x2 filter.

Convolution mode

Cross-correlation for forward, arbitrary for **`dgrad`** and **`wgrad`**.

Dilation

The dilation is 1x1x1.

Platform

The platform is Volta with input/output channels divisible by 8.

Batch/image size

cuDNN will fallback to non-Tensor Core kernel if it determines that the workspace required is larger than 256MB of GPU memory. The workspace required depends on many factors. For the Tensor Core kernels, the workspace size generally scales linearly with output tensor size. Therefore, this can be mitigated by using smaller image sizes or minibatch sizes.

2.1. Recommended Settings In cuDNN While Performing 3D Convolutions

The following table shows the specific improvements that were made in each patch release.

Table 1 Recommended settings while performing 3D convolutions

Volta						
cuDNN version		**7.6.2**	**7.6.2**	**7.6.1**	**7.6.1**	**7.6.1**
Convolution (3D or 2D)		3D				
Convolution or deconvolution (`fprop`, `dgrad`, or `wgrad`)		`dgrad`	`fprop`	`wgrad`	`dgrad`	`fprop`
Grouped convolution	Yes or No	No				
	Group size	NA				
Data layout format (`NHWC`/`NCHW`)[1]		NCDHW	NCDHW[2]			
Input/output precision (FP16, FP32, or FP64)		FP16 or FP32	FP16[3] or FP32[4]			
Accumulator (compute) precision (FP16, FP32, or FP64)		Better to be the same input and output precision	FP32			
Filter (kernel) sizes		2x2x2	• T[5]x1x1 • Tx2x2 • Tx3x3 • Tx5x5	• 1x1x1 • 2x2x2 • 3x3x3 • 5x5x5	• Tx1x1 • Tx2x2 • Tx3x3 • Tx5x5	• Tx1x1 • Tx2x2 • Tx3x3 • Tx5x5
Padding			Filter // 2[6]			
Image sizes			256 MB WS limit[7]	256 MB WS limit[8]	256 MB WS limit[9]	256 MB WS limit[10]

1 `NHWC`/`NCHW` corresponds to `NDHWC`/`NCDHW` in 3D convolution.
2 With NCHW <> NHWC format transformation.
3 FP16: `CUDNN_TENSOROP_MATH`
4 FP32: `CUDNN_TENSOROP_MATH_ALLOW_CONVERSION`
5 An arbitrary positive value.
6 `padding = filter // 2` constraints is no longer required in integrated kernel
7 `fprop`: reduction

```
buffer size = ceil(k / tileN) * tileN * ceil(n*o*p*q / tileM) *
    tileM
```

8 `wgrad`: reduction

```
buffer size = ceil(c / tileN)* tileN * ceil(k*t*r*s /
```

Volta						
cuDNN version		**7.6.2**	**7.6.2**	**7.6.1**	**7.6.1**	**7.6.1**
Number of channels	C	Arbitrary	0 mod 8			
	K	Arbitrary	0 mod 8			
Convolution mode			Cross-correlation			Cross-correlation
Strides		2x2x2	Arbitrary stride	1x1x1		
Dilation		1x1x1				

Chapter 3. MEDICAL IMAGING PERFORMANCE

The following table shows the average speed-up of **unique cuDNN 3D convolution calls** for each network that satisfies the conditions in Best Practices For Medical Imaging.

Model	Batchsize	Avg. Speed-up of unique cuDNN 3D convolution API calls (7.6.3 vs. 7.5.1)
V-Net (3D-Image segmentation)	2	4.4x
	4	4.4x
	8	4x
	16	4x
	32	4x
	64	3.4x
	128	3x
3D-UNet (3D-Image Segmentation)	2	4.4x
	4	4.1x
	8	4.4x
	16	4.3x
	32	4x
	64	4x
	128	4.2x

Chapter 4. MEDICAL IMAGING LIMITATIONS

Your application will be functional but slow if the model has:

- Channel counts lower than 32 (gets worse the lower it is)
- Data gradients for convolutions with stride

If the above is in the network, use **`cuDNNFind`** to get the best option.

Notice

Trademarks

Copyright

www.nvidia.com