Performance Guide for ViDi

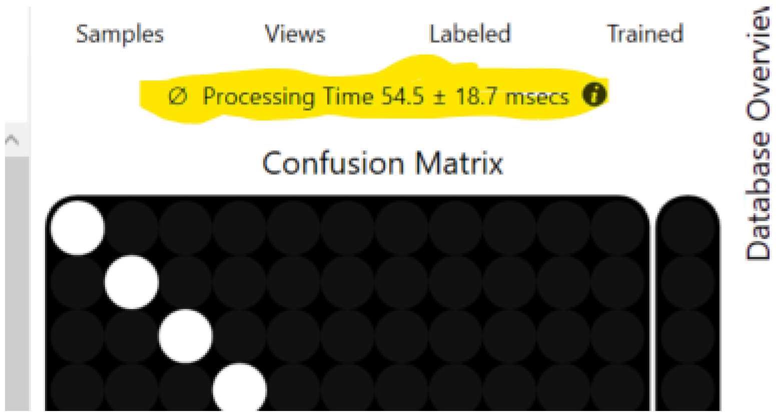
18-Mar-2019 16:11:24 EDT

Thinking about Performance

What performance aspect is important to you?

Tool and stream processing time

Individual tool processing time is shown in the Database Overview panel:



The reported time is the average processing time for all of the images processed during the most recent processing.

The processing time for a stream containing multiple tools is not available through the ViDi Suite GUI, and you cannot estimate it by summing the tool execution time, as it includes the time required to prepare and transmit view information between tools.

When thinking about stream processing, remember that the processing of tools in a ViDi stream is always serialized when you call **Stream. Process()**. Only one tool is ever processed at a time unless you explicitly process tools individually using **Tool.Process()**.

Throughput

Throughput refers to the total number of images that can be processed per unit time. If your application can process multiple streams concurrently using different threads, it may be able to improve system throughput, although individual tool processing will be slower.

Performance Toolkit

In terms of increasing expense (but not efficacy):

- Application design
- Tool parameters
- System configuration
- Hardware options
- Multiple GPUs

Application Design

The following table summarizes application design characteristics that may produce faster run-time performance. Application design choices that improve performance typically have minimal impact on the behavior of the system.

Design Pattern	Why it's Faster	Best Bang for Buck	But Watch Out For
Use a small number of tools per stream.	The processing time for a single ViDi tool does not vary significantly based on the amount of information that the tool returns. For example, a single Blue tool that is trained to find 100 different features runs at the same speed as a tool that is trained to find only a single feature. Further, the number of features returned makes only a minimal speed difference. Similarly, a Red tool runs at the same speed regardless of how many defects it finds, and a Green tool can classify into 2 classes or 2000 classes at the same speed.	Start building your application with a single tool.	
Avoiding Image Conversions	During tool operation, the image must be sampled for processing by the neural network. This sampling requires a raster (uncompressed) format image such as a bitmap. Performing this conversion takes time. Similarly, if the tool is configured to use a single-channel (grey-scale) image as input, but the supplied images are multi-channel color images, the luminance value must be computed for each image at run time.	Attempt to solve your application using a single-channel grey-scale image.	Some applications require color information.
Reduce the amount of processed data	Reducing the number of processed data by: • Using a smaller ROI • Using a mask • Using as few image channels as possible Improves processing speed by reducing the total amount of data processed.	Restricted ROI	ViDi tools need contextual information to work well – don't constrain the ROI too much. Downsampling is usually not needed – selecting a larger feature size can improve speed and remove the need for run-time downsampling.
Multi-threading	On systems with multiple GPUs, processing multiple streams concurrently allows tools to execute in parallel, increasing throughput. On single-GPU systems, you can configure the system to allow multiple processes to make use of the same GPU. This allows a higher GPU occupancy and can improve throughput, although tool execution time will increase.	Use the **--max-process-count** command line argument to enable multiple threads to access a single GPU. To enable multi-process GPU access for a runtime application using the local control's **GlobalConfig()** method: **`control.GlobalConfig(`** **"max_process_count=2");**	Processing time for an individual tool will increase.
C++ (unmanaged)	Use of an unmanaged language environment reduces the impact of system activity on tool execution.	For low-latency, high-performance applications, use the C++ API.	Windows is not a RTOS

Tool Parameters

Tool parameter choices directly effect tool execution speed, but there is typically a tradeoff between tool speed and accuracy or robustness.

Tool Parameter	How it Affects Speed	Best Bang for Buck	But Watch Out For
Feature size	At run time, ViDi tools need to sample the entire input image. The feature size determines the number of samples required for a given image size. The larger the feature size, the fewer the samples.	$O(n^2)$ increase in speed with larger feature size.	Larger feature sizes may cause the tool to miss features or defects. Use parameter optimization to find an optimal size.
Sampling Density	Similarly to feature size, the sampling density determines the number of samples required for a given image size.	$O(n^2)$ increase in speed with lower sampling density.	Risk of missing features or defects.
Refinement Parameters	The Blue and Red tools include processing-time parameters that provide more accurate results at the cost of increasing execution time: • Blue tool: **Precision** • Red tool: **Iterations**	Increasing the iteration value increases processing time linearly.	
Low-precision mode	If your system meets certain specific requirements (CUDA Compute Capability 6.1 or greater), you can enable **low-precision processing** mode for any ViDi tool. Enabling low-precision mode converts any existing trained tool to use low-precision computation during processing, and it generates low-precision tools for all future training operations until it is disabled. (Once a tool has been converted to low-precision mode, it must be retrained to disable low-precision mode. Low-precision tools can execute from 25% to as much as 50% faster than normal-precision tools.	Additional run-time speed improvements for low-precision tools are seen on systems with Turing Tensor cores.	Changing a tool to low-precision mode may change the results the tool produces to a small degree. Generally high-level feature identification, defect classification, and general classification will be unchanged, but specific feature and defect regions and scores may change slightly.

NVIDIA GPU Selection and Configuration

System configuration choices directly affect tool processing speed without affecting tool accuracy or behavior. They are the most expensive and hardest to predict the effect of.

Configuration Option	Why it's Faster	Best Bang for Buck	But Watch Out For
NVIDIA Device Type	The number of CUDA cores is directly related to high-precision processing speed and training. The number of standard Tensor cores is directly related to processing speed and training speed. The number of **Turing** Tensor cores is related to processing speed in low-precision mode only. These cores do not affect standard precision processing or training speed.		

NVIDIA Driver Mode	Consumer-grade gaming-oriented NVIDIA devices only support the WDDM device driver model. This driver is intended to support graphics display, not computation. Professional-grade NVIDIA cards support the TCC driver mode, which provides better performance and stability.	Select a Quadro or Tesla (or selected Titan)-branded NVIDIA card.	If using a GeForce-branded card, be aware that NVIDIA Geforce drivers are updated frequently and may not be compatible with ViDi. Please visit Cognex's support page for driver recommendations. Using TCC mode driver prevents the use of Video output on the GPU card; use onboard video instead.
Optimized memory	ViDi optimized memory, which is enabled by default, improves performance by overriding the standard NVIDA GPU memory management system.	Make sure your card has at least 4GB of GPU memory. Performance improvement is not as significant for cards using the TCC driver.	

NVIDIA Device Branding Summary

The following table summarizes the different NVIDIA device types.

Class	**Consumer**		**Professional**	
Branding	**GeForce**	**Titan**	**Quadro**	**Tesla**
Volta Architecture Cards	---	Titan V	GV100	V100
Pascal Architecture Cards	GTX 1xxx	Titan Xp	G/GPxxx	P100
Turing Architecture Cards	RTX 2xxx	Titan RTX	Quadro RTX4xxx	T4
Video Output	**Yes**	**Yes**	**Yes**	**---**
Price Point	$1K	$3K	$5K	$5K+
TCC Driver Support	**---**	**Yes**	**Yes**	**Yes**
ECC Memory	**---**	**---**	**Yes**	**Yes**
Tensor Cores	**Yes**: RTX2xxx and newer	**Yes**: Titan V **Yes**: Titan RTX	**Yes**: Quadro RTX **Yes**: Quadro GV100	**Yes**: V100 **Yes**: T4

Graphics Card Requirements

- NVIDIA® CUDA® enabled GPU
- CUDA compute capability 3.0 or higher

Considerations

While consumer cards and professional cards perform similarly, some considerations should be made:

- Heat dissipation
  - Professional cards are intended for continuous duty cycle use and are designed to dissipate heat effectively.
- Supply
  - Professional cards are manufactured by NVIDIA and have a longer product cycle.
- Performance and Control
  - Professional cards support the TCC mode drivers. This allows the GPU to run as a computing device with no display output.
    - This means you will need a second card for display (or use the motherboard's built-in display).

Estimating Run-Time Performance

The following numbers are an approximate guide to the potential **run-time** performance increment for different card families (baseline = non-TensorCore, standard mode):

ViDi Operating Mode	No Tensor Cores (ex GTX)	Volta Tensor Cores (ex V100)	Turing Tensor Cores (ex T4)
Standard	100%	150%	150%
Low-precision	125%	125%	175%

Note

In comparison with other Tesla cards, the T4 is oriented toward run-time operation. It supports ViDi training and run-time, but training performance will likely be slower than a V100.

Glossary of Standard NVIDIA GPU Terminology

Term	What it is	Is it important?
CUDA Core	Standard NVIDIA parallel processing unit.	**Yes**. This is the 'standard' measure of NVIDIA GPU processing – the number of CUDA cores. The more cores, the faster the ViDi processing and training.
ECC Memory	Error-correcting-code memory Hardware support for verifying that memory reads/writes do not contain errors.	**No** Because of the huge number of computations involved in training and processing neural networks, the likelihood of a memory error affecting a tool result is very low.
TCC	Tesla Compute Cluster (Driver). A high-performance driver that is optimized for computational use of an NVIDIA GPU. • Not supported by all cards • Disables video output from the card • Provides faster training and runtime performance • Diminishes or eliminates the advantages of using ViDi optimized memory • Configured using the nvdia-smi utility	**Yes**. Whenever possible, customers should select cards that support the TCC driver mode, and they should enable the mode.
Tensor Core	Full-precision, mixed-precision (and evt. integer math) parallel processing unit dedicated to matrix multiply operations.	**Yes**. Starting with ViDi 3.2, ViDi automatically takes advantage of tensor cores for faster processing and training, as long as the user has a Standard or Advanced license.
TensorRT	NVIDIA framework for optimizing (by using low-precision and integer math) run-time performance of TensorFlow, Caffe, and other standard framework networks running on a GPU with Tensor Cores.	**No:** ViDi uses a proprietary network architecture that is not compatible with Tensor RT.

Multiple GPUs

Except under very narrow circumstances, using multiple GPUs in a single system will not reduce ViDi tool training or processing time. What multiple GPUs can do is to:

- Increase system throughput when your application uses multiple threads to concurrently process images
- Increase training productivity, by allowing you to train multiple tools at the same time

There is one circumstance under which multiple GPUs can be used to reduce tool processing time. If you configure your system in **MultipleDevic esPerTool** mode, then all installed GPUs are treated as a single GPU during processing. This means that only one tool can be processed at a time for the entire server.

In the specific case of a Red Analyze tool, the use of **MultipleDevicesPerTool** mode may speed up the tool, especially a tool with a high image-to-feature size ratio. However, this potential speed up comes at the expense of latency across all clients.

System Configuration for Multi-GPU Systems

When configuring a host system for multiple GPUs, keep the following in mind:

- The chassis may need to provide up to 2KW of power
- Quadro and Tesla cards provide better cooling configuration for multiple-card installations
- Make sure that the PCIe configuration has 16 PCIe lanes available for each GPU
- Do not enable SLI

What About Training Time?

Reducing tool training time does not affect your performance at run time, but it can improve the productivity of your development team.

ViDi training uses a mixture of CPU and GPU resources. When considering training specifically, there are three phases: computing image statistics, building the model, and then processing the image set with the newly trained model. The model building phase of training usually takes the longest, and it is an iterative process. Each iteration requires that the tool generate training data from all of the training images. If the images are in a non-BMP format, they need to be converted to BMP for each iteration.

- Tool training is always single-threaded and single GPU. You cannot make training faster using multiple GPUs.
- Using multiple GPUs can improve your productivity because you can train multiple tools concurrently.