TIRIAS RESEARCH

PLASTER:
A Framework for Deep Learning Performance

Whitepaper sponsored by NVIDIA

David A. Teich, Paul R. Teich
TIRIAS Research
May 2018

Introduction

Machine learning (ML) is a key category in artificial intelligence (AI). Both hardware and software advances in deep learning (DL), a type of ML, appear to be catalysts for the early stages of a phenomenal AI growth trend. The challenge at this phase of adoption is twofold: deploying deep learning solutions is a complex proposition, and it is a rapidly moving target. The industry needs a framework to address the opportunities and challenges associated with deep learning.

At the NVIDIA GPU Technology Conference (GTC) 2018, Jensen Huang, NVIDIA President and CEO, put forward the PLASTER framework to contextualize the key challenges delivering AI-based services (Figure 1).

Figure 1: PLASTER Framework for AI

Source: NVIDIA

"PLASTER" encompasses seven major challenges for delivering AI-based services.

- **P**rogrammability
- **L**atency
- **A**ccuracy
- **S**ize of Model
- **T**hroughput
- **E**nergy Efficiency
- **R**ate of Learning

This paper explores each of these AI challenges in the context of NVIDIA's DL solutions. PLASTER as a whole is greater than the sum of its parts. Anyone interested in developing and deploying AI-based services should factor in all of PLASTER's elements to arrive at a complete view of deep learning performance. Addressing the challenges described in PLASTER is important in any DL solution, and it is especially useful for developing and delivering the inference engines underpinning AI-based services. Each section of this paper includes a brief description of measurements for each framework component and an example of a customer leveraging NVIDIA solutions to tackle critical problems with machine learning.

Programmability

Machine learning is experiencing explosive growth not only in the size and complexity of the models but also the burgeoning diversity of neural network architectures. It is difficult even for experts to understand the model choices and then choose the appropriate model to solve their AI business problems.

After a deep learning model is coded and trained, it is then optimized for a specific runtime inference environment. NVIDIA addresses training and inference challenges with two key tools. For coding, AI-based service developers use CUDA, a parallel computing platform and programming model for general computing on GPUs. For inference, AI-based service developers use TensorRT, NVIDIA's programmable inference accelerator.

CUDA helps data scientists by simplifying the steps needed to implement an algorithm on the NVIDIA platform. The TensorRT Programmable Inference Accelerator tool takes a trained neural network and optimizes it for runtime deployment. It tests different levels of floating point and integer precision, so that developers and operations can balance system-required accuracy and performance to provide an optimized solution.

Developers can use TensorRT directly from within the TensorFlow framework to optimize models for AI-based service delivery. TensorRT can import Open Neural Network Exchange (ONNX) models from a variety of frameworks, including Caffe2, MXNet, and PyTorch. While deep learning is still coding at a technical level, this will help the data scientist better leverage valuable time.

Measuring Programmability

Programmability affects developer productivity and therefore time-to-market. TensorRT accelerates AI inference on multiple popular frameworks, including Caffe2, Kaldi, MXNet, PyTorch, and TensorFlow. In addition, TensorRT can ingest CNNs, RNNs and MLP networks, and offers a Custom Layer API for novel, unique, or proprietary layers, so developers can implement their own CUDA kernel functions. TensorRT also supports the Python scripting language, allowing developers to integrate a TensorRT-based inference engine into a Python development environment.

Programmability in Action

Baker Hughes (BHGE) is a leading oil field services company. It helps oil and gas companies in all aspects of exploration, extraction, processing, and delivery. At each step of the process, AI can help oil and gas companies better understand the massive volumes of data their operations create. Each type of business need can lean on a different type of deep learning model. That means programmers must efficiently be able to implement, test, and instantiate multiple models.

BHGE uses CUDA and TensorRT to create the deep learning models that help its customers identify and locate oil and gas resources. BHGE also uses NVIDIA hardware, including DGX-1 servers for model training; DGX Stations at the deskside or on remote offshore platforms for

model training and inference: and NVIDIA's Jetson platform for real-time, continuous deep learning, and inferencing at the edge of the internet of things (IoT).

Latency

Humans and machines need a response to make decisions and take action. Latency is the time between requesting something and receiving a response. With most human-facing software systems, not just AI, the time is often measured in milliseconds.

Voice recognition is a commonly understood application, thanks to Siri, Alexa, and similar voice interfaces. There is wide demand for digital assistants in both consumer and customer service applications. But when humans try to interface with digital assistants, a lag of even a few seconds starts to feel unnatural.

Image and video management is another example of the need for low latency, real-time inference-based service needs. Google has stated that 7 milliseconds is an optimal latency target for image-and video-based uses.

Another example is automatic translation. Earlier systems, based on a more programmatic, expert system design, were unable to understand the nuances of language fast enough to provide realistic conversation. Now DL is doing a far better job creating far improved translations.

Measuring Latency

Inference latency directly affects user experience (UX) and is measured in seconds or fractions of a second. While there are no strict laws for response times, Jakob Nielsen's 0.1 / 1 / 10 second limits are a good guideline. Somewhere between 2 to 10 seconds, people start wondering if the system is still working properly. Users lose the flow of their activities, and that impacts enjoyment, performance, time, and money.

Latency in Action

Visual search continues to emerge as a new frontier for online searches. Microsoft's Bing server group was looking to deliver a visual search platform that could quickly deliver search results, and it has built a neural network-based solution. The initial system latency was around 2.5 seconds, but using Tesla GPUs, Microsoft reduced the latency to just 40 milliseconds, netting a 62x improvement.

Accuracy

While accuracy is important in every industry, healthcare needs especially high accuracy. Medical imaging has advanced significantly in the last couple of decades, increasing usage and requiring more analysis to identify medical issues. Medical imaging advancements and usage also mean that large volumes of data must be transmitted from medical machines to medical specialists to analyze. Options to address the data volume issue have been either to transmit the full information with long delays or to sample the data and reconstruct it using techniques that can lead to inaccurate reconstruction and diagnostics.

An advantage of deep learning is that it can be trained at high precision and implemented at lower precision. DL training can occur very accurately at a higher level of mathematical precision, usually FP32. Then implementation in runtime can occur with lower precision math. often FP16, gaining improved throughput, efficiency, and even latency. Maintaining high accuracy is essential for best user experiences. TensorRT takes advantage of the Tesla V100 Tensor Core's FP16 processing, as well as Tesla P4's INT8 feature to accelerate inference by 2-3x compared to FP32 with near-zero loss in accuracy.

AI-based service developers can optimize their deep learning models for efficiency and then affordably implement them in the field.

Measuring Accuracy

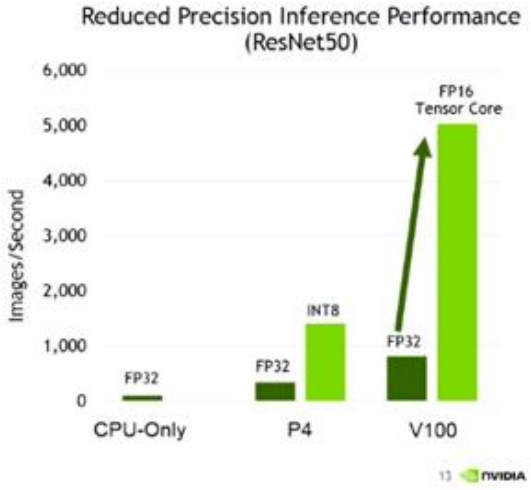
There are many ways to define measurements of accuracy. For PLASTER, accuracy is a question of retaining trained model accuracy in runtime inferences, while also optimizing inference performance for runtime efficiencies (improved latency, *etc*.). The key to accuracy at runtime is lowering the mathematical precision to gain power and energy efficiency, increasing throughput, and leveraging other benefits without accuracy falling below the necessary level for each application. TensorRT helps with the accuracy decision by allowing inference at multiple levels of precisions to compare change in accuracy (Figure 2).

Figure 2: TensorRT Reduced Precision Inference Performance

	FP32 Top 1	INT8 Top 1	Difference
Googlenet	68.87%	68.49%	0.38%
VGG	68.56%	68.45%	0.11%
Resnet-50	73.11%	72.54%	0.57%
Resnet-152	75.18%	74.56%	0.61%

Precision calibration for INT8 inference:

- Minimizes information loss between FP32 and INT8 inference on a calibration dataset
- Completely automatic



Source: NVIDIA

Accuracy in Action

Massachusetts General Hospital's AA Martinos Center for Biomedical Imaging and Harvard University are working on systems to speed and improve MRI image reconstruction. Leveraging a DGX-1, they have created AUTOMAP, a deep learning system used to reconstruct images directly from sensor data. This deep learning system filters out noise and defects to reconstruct images 100x faster and with 5x higher accuracy for more accurate diagnostic outcomes.

Size of Model

The size of a deep learning model and the capacity of the physical network between processors have impacts on performance, especially in the latency and throughput aspects of PLASTER. Deep learning network models are exploding in numbers. Their size and complexity are also increasing, enabling far more detailed analysis and driving the need for more powerful systems for training. In a deep learning model, the drivers of compute power and physical network expansion are:

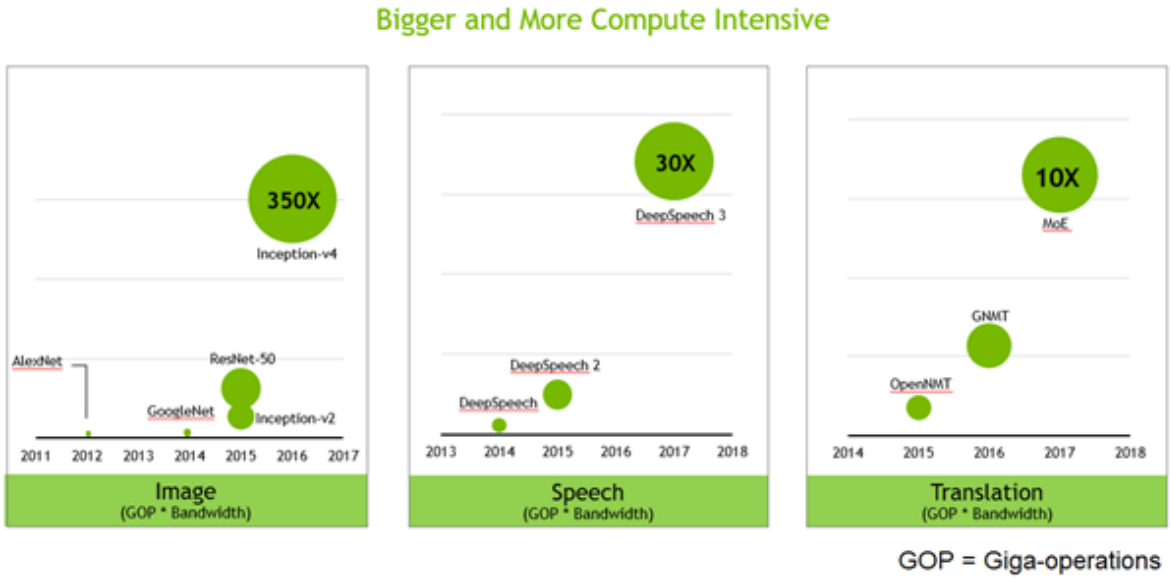
- Number of **layers**
- Number of **nodes** (neurons) per layer
- **Complexity** of computation per layer
- Number of **connections** between a node at one layer and the nodes of neighboring layers

It is early in the DL market lifecycle. When comparing model size, current thinking boils down to a practical relationship: DL model size is proportional to the amount of compute and physical networking resources needed to run inferences, within the context the other PLASTER components. For example, when a developer optimizes a trained DL model to stay within inferencing accuracy and latency bounds, that optimization may reduce computational precision, simplify each model layer, and simplify the connectivity between model layers. However, starting with a larger trained model usually results in a larger optimized model for inferencing.

Measuring Size of Model

Developers often describe DL model size in terms of compute demand and memory latencies. For deep learning models, Figure 3 shows that size is a combination of both compute demand and the physical network bandwidth needed to move data in the computational memory space.

Figure 3: Deep Learning Model Size



Source: NVIDIA

Deep learning models across several major usage areas have grown by 1-2 orders of magnitude. This growth in size, complexity, and computational demand, coupled with the emergence of real-time services requiring low latency, highlights the challenge of model size. The impact on latency and throughput values for larger models must be addressed at the hardware level and with adjustments to runtime inference accuracy (precision).

Size of Model in Action

Baidu Research released its original Deep Speech (DS1) speech recognition model in late 2014. DS1 uses a 5-layer convolutional neural network (CNN) model with 1 recurrent neural network (RNN) layer and about 8.1 million parameters. A year later, the second generation Deep Speech 2 (DS2) model used a 7-layer RNN with 3 layers of CNN and about 67.7 million parameters, an 8.3x increase over DS1. Recognition word error rate (WER) decreased from 24.0% using DS1 to 13.6% using DS2, a 43% improvement. The increase in size and complexity of the newer DS2 model was directly responsible for the improvement in speech recognition accuracy. NVIDIA and Baidu announced a partnership in 2017 to accelerate both AI training and acceleration in the datacenter.

Throughput

Throughput describes how many inferences can be delivered given the size of the deep learning network created or deployed. Developers are increasingly optimizing inference within a specified latency threshold. While the latency limit ensures good customer experience, maximizing throughput within that limit is critical to maximizing datacenter efficiency as well as revenue.

There has been a tendency to use throughput as the only performance metric, as more computations-per-second generally leads to better performance across other areas. However, if a system cannot deliver adequate throughput within a specified latency requirement, power budget, or server node count, then the system will not ultimately serve an application's inference needs well. Without the appropriate balance of throughput and latency, the result can be poor customer service, missing service level agreements (SLAs), and potentially a failed service.

The entertainment industry has long used throughput as a key performance indicator, especially in dynamic ad placement. For example, brand sponsors dynamically place ads into streaming video such as television programs or sporting events. Advertisers want to know how often their ads are showing up, and if they are reaching their intended audience. Reporting to advertisers on the accuracy and focus of those placements is critical to keeping the advertisers happy.

Measuring Throughput

DL inference throughput is generally expressed as images-per-second for image-based networks and tokens-per-second for speech-based networks. The system must achieve throughput within a specified latency threshold. Service operators can manage inferencing throughput by scaling the number of GPUs to maintain the appropriate latency for each inference. This scaling is possible if adding more GPUs does not increase latency over the first GPU.

Throughput in Action

Audi sponsors many sporting events. As an SAP key customer, it received early access to SAP Brand Impact, SAP's DL solution for tracking ad placements (Figure 4).

Figure 4: Image Identification During Live Programming

Source: NVIDIA

Audi developed its deep learning model with SAP Brand Impact. Audi trained its model on NVIDIA DGX-1 servers using CUDA and then optimized its model for inference using TensorRT. The result is a 40x performance improvement versus a CPU-only solution, a 32x reduction in hourly costs. Results are immediate, accurate, and auditable, allowing SAP Brand Impact to provide results to their customers while programs are still broadcasting.

Energy Efficiency

As DL accelerator performance improves, DL accelerator power consumption escalates. Providing ROI for deep learning solutions involves more than looking at just the inference performance of a system. Power consumption can quickly increase costs of delivering a service, driving a need to focus on energy efficiency in both devices and systems.

Speech processing is a good example of a solution that needs heavy processing to provide an intelligent response in a natural voice. Datacenter inference providing real-time processing for speech can easily involve large racks of machines that can impact a company's total cost of ownership (TCO). Therefore, the industry measures operational success in inferences-per-watt (higher is better). Hyperscale datacenters seek to maximize energy efficiency for as many inferences as they can deliver with a fixed power budget.

The solution is not as simple as looking at which individual processor has lower power consumption. For instance, if one processor is pulling 200W vs. another pulling 130W, that does not necessarily mean the 130W system is better. If the 200W solution completes the task 20x faster, it is more energy efficient.

Inferences-per-watt also depends on the latency factors in both training and inference. Energy efficiency depends not only on the pure power consumption over time, but the throughput during the same time. Energy efficiency is another example illustrating how PLASTER's elements are interrelated and must be considered together for a complete inference performance picture.

Measuring Energy Efficiency

For production inference, energy consumption constrains available compute resources. Scaling a cloud service to handle more inferences per second while maintaining quality depends directly on the operational expenses (OPEX), such as power and cooling. Scaling endpoint inference capability may affect the cost and weight of batteries in the endpoint, the number and quality inferences delivered between battery charges, *etc*.

Energy Efficiency in Action

IFLYTEK is China's leader in speech recognition, with more than 910 million users. IFLYTEK's cloud-based speech recognition service must respond to users within 200 ms to give users a realistic and natural voice response experience. Its inferencing service runs on servers that host NVIDIA Tesla P4 GPU cards. IFLYTEK's cloud-based speech recognition inference-as-a-service can now handle 10x more concurrent requests than with CPU-only servers. The system handles 10x more requests with 20% better accuracy and a 20% reduction in operational TCO. The TCO reduction is due in large part to improved inferences-per-watt performance.

Rate of Learning

In recent years, businesses have started implementing DevOps to tie development and operations more tightly, aided by more powerful systems and higher-level programming tools. While deep learning is still nascent, the many academic, governmental, and business institutions waiting to leverage DL are not. They are not looking for an inference engine that has been trained in a void and remains static. As one of the two words in "AI" is intelligence, users will want the neural networks to learn and adapt in a reasonable timeframe. For complex DL systems to gain traction in business, software tool developers must support the DevOps movement.

As organizations continue to experiment with deep learning and neural networks, they are learning how to more effectively build and implement DL systems. DL models must be retrained periodically as inferencing services gather new data and as services grow and change. Therefore, IT organizations and software developers must increase the rate at which they can retrain models as new data arrives. Multi-GPU server configurations have reduced deep learning training times from days and weeks to minutes and hours. Faster training times mean that developers can retrain their networks more often to improve accuracy or maintain high accuracy. Some deep learning implementations today already retrain their neural networks multiple times every day.

Programmability also factors into rate of learning. To reduce developer workflow, Google and NVIDIA recently announced an integration between TensorFlow and TensorRT. Developers can invoke TensorRT from within the TensorFlow framework to optimize those trained networks to

run efficiently on NVIDIA GPUs. The ability to integrate training and inference more easily enables deep learning as a DevOps solution, helping organizations rapidly implement change as they evolve their DL models.

Measuring Rate of Learning

Rate of learning is measured in the following ways:

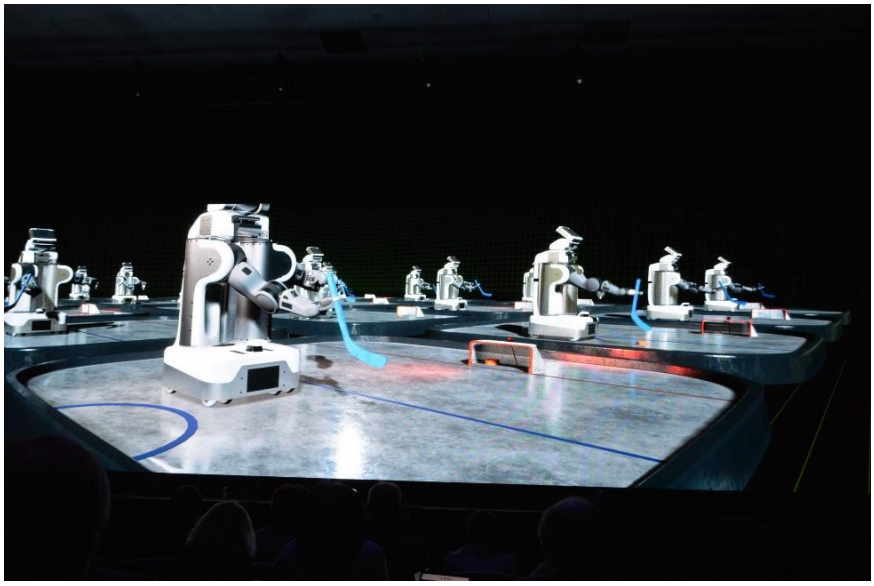
- For training: Improvements in throughput and model accuracy
- For production: Improvements in throughput, model accuracy, and latency
- For both training and production: Improvements in programmability, size of model, and energy efficiency

DevOps helps DL organizations maximize these improvements by embracing and integrating new deep learning advancements.

Rate of Learning in Action

NVIDIA has created a simulation environment called Isaac Sim to train its AI-enabled robot Isaac (Figure 5). NVIDIA has demonstrated this environment using its Project Holodeck, a collaborative and physically accurate virtually reality (VR) environment. In this demo, virtual robots are learning how to play hockey. Each time one robot completes an iteration, its learnings are propagated to the other virtual robots to accelerate the overall learning process.

Figure 5: NVIDIA Isaac Sim



Virtual robots learning in the Isaac Sim environment
Source: NVIDIA

Developers can use Isaac Sim to train and test virtual robots using fully integrated and high-fidelity simulation environments and do so in faster-than-real-world speeds. Engineering and testing happen in minutes instead of months. Once a simulation is complete, the trained system (brain) can be transferred to physical robots.

PLASTER: A Framework for Deep Learning Performance

Deep learning is moving from theory into early-stage adoption. The industry must evaluate what it will take to support DL as it moves towards mainstream. The rapid evolution of algorithms, software, and hardware to support DL requires a framework to understand the challenges and create a context to meet those challenges.

PLASTER originated as an NVIDIA's rubric to describe the key elements of deep learning performance. The components of PLASTER are: Programmability, Latency, Accuracy, Size of model, Throughput, Energy efficiency, and Rate of learning. Good DL project design considers all these factors to arrive at the right set of tradeoffs necessary to produce a successful DL implementation. The components of PLASTER define the industry's challenges in creating DL solutions. None of the components is independent, and all are important.

The PLASTER framework is important for the entire DL model development and deployment cycle. PLASTER is particularly important for runtime inference. Production has specific parameters that must be met, from the basics of accuracy and performance that drive ROI, to meeting SLAs for latency and throughput, to the needs for regulatory and corporate compliance.

Organizations that consider PLASTER as an organizing principle can achieve three outcomes that will advance DL towards the mass market. DL organizations can

- Better manage performance of the DL systems
- Make more efficient use of developer time
- Create a DevOps environment in DL to support the products and services customers want

Deep learning is a complex problem in the early stages of its product lifecycle. Organizations that use PLASTER as a framework can better understand and manage the critical aspects of deep learning performance.