Can real-time Artificial Intelligence techniques be applied to Synthetic Environments?

Graham Long
Thales UK
Crawley, West Sussex
Graham.Long@uk.thalesgroup.com

ABSTRACT

The increasing proliferation of source data poses a significant challenge to the capabilities of Synthetic Environment (SE) generation pipelines to transform this data with the necessary agility, velocity, productivity and affordability required to meet future demands.

Previous research has demonstrated the potential to apply Artificial Intelligence (AI) throughout the entire generation pipeline. But the ability to capture large volumes of imagery, video, point cloud and other real-world data within days, hours or in real-time places specific demands on the capacity of SE pipelines to extract and process features, attributes and properties from this data, and update or create SE's at a comparable tempo.

Faced with a similar and potentially overwhelming data glut, fields such as security, autonomous vehicle and Intelligence, Surveillance and Reconnaissance (ISR) systems are performing automated, real-time analysis and computer vision tasks, from facial recognition and human pose estimation to object detection, classification and segmentation. These are made possible by the application of AI, particularly Deep Learning (DL), to perform critical data processing and analysis in real-time, often on edge devices.

If similar techniques can be applied to SE generation it offers a path to genuinely rapid SE construction. Today's largely offline data preparation and processing tasks could be encapsulated into a pipeline that can intelligently analyze, process and exploit input data, extract features, derive and generate content on-the-fly, in an end to end, data to run-time SE construction process.

This paper will examine the state and application of AI to rapid or real-time data processing and analysis. It will assess if, how and where such implementations of AI and DL could be applied to introduce similarly fast, dynamic data exploitation into the SE pipeline, and consider potential issues such solutions may pose in areas such as SE interoperability.

ABOUT THE AUTHORS

Graham Long has worked in the simulation industry for over 30 years. Starting his career as a synthetic environment developer, he has managed engineering teams engaged in the deployment of image generator and display systems, synthetic environment production and the development of synthetic environment generation and authoring tools, delivering solutions to over 20 civil and military flight simulation projects. His responsibilities at Training Solutions, Thales (UK), incorporate synthetic environment development, generation technologies, solutions, tools and interoperability standards.

Can real-time Artificial Intelligence techniques be applied to Synthetic Environments?

Graham Long
Thales UK
Crawley, West Sussex
Graham.Long@uk.thalesgroup.com

INTRODUCTION

Synthetic Environments (SE) provide a representation of our real physical world and are dependent upon data describing the real world as the foundation for SE construction. Fleets of small, low cost mini satellites now capture images of the entire earth every day. Drones have commoditised data collection, enabling on-demand data capture and ease of data access to many users, markets and applications. The range of exploitable data types is diversifying. From multi-resolution imagery, LiDAR (Light Detection and Ranging), and Full Motion Video (FMV) - not to mention the estimated 20billion "Internet of Things" (IoT) devices capturing a vast array of information from climate data to imagery and the enormous Social Media data lake - both offer a new and potentially valuable but currently underexploited data source.

The agility and velocity at which this data is expanding and changing can occur in real-time, hours, or days. It is in stark contrast to the rate of most SE development and updates that often happen over weeks, months, and even years. Not all SE are supporting use cases with time critical construction or data currency requirements, but static SE that lag and no longer reflect the state of the real-world are of diminishing value. This is particularly true for today's increasingly connected, distributed SE systems that must support the needs of multiple SE clients, platforms and domains, perhaps operating in a Live, Virtual, Constructive (LVC) context.

SE construction pipelines must adapt to address this situation, or they will be overwhelmed by the volume and rate of data input. Too slow to process data, too slow to generate the SE, they risk becoming unable to fully exploit the data's value and swiftly produce an SE of acceptable fidelity and quality.

SE Generation Today

The Synthetic Environment pipeline is moving toward architectures that publish or render a synthetic environment directly from source data by streaming or paging this data directly to the synthetic environment client (Figure 1). This eradicates the need to generate an intermediate, off-line version of the environment, increasing pipeline agility. Almost instantaneous updates of the real-time SE in response to source data changes may simply involve updating the relevant data tiles on a terrain data server (with new image tiles for example).

While this approach may help improve responsiveness to rapidly changing real-world data, it does not overcome the fundamental fact that in most cases, source data has to undergo varying degrees of processing and preparation for SE use before it gets to the data server. And in practise, data layers are related – imagery, transport networks, buildings, vegetation, materials, 3D features – and can be rarely updated or changed independently. For example, it is very probable new or updated imagery will require updates to road networks, building or vegetation data layers. Updating multiple data layers often becomes a significant time and cost driver – are these layers available as existing data products? are they current and correlated to the imagery? If not, they will need to be created, which will typically require extraction and derivation from the imagery. Without highly automated processes, performing these processes at scale can be prohibitively inefficient – for example, manually capturing all footprints in a city of 200,000 buildings in 500hrs would require an optimistic capture rate of 6.5 buildings a minute. Responding to new and changing data inputs can quickly escalate into sizeable and complex data and SE processing issues.

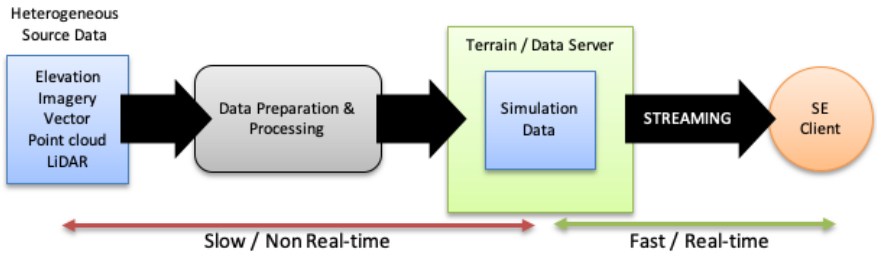

Figure 1. Notional SE Pipeline Architecture

These issues can be mitigated with various approaches. Volunteered Geographic Information (VGI) such as Open Street Map is commonly used in many SE pipelines as a ready-made global source of cultural features. But it is not a perfect solution. Its content, coverage, currency and quality are very inconsistent, and it cannot be relied upon as an off-the-shelf data source. It is also unlikely to fully correlate with imagery or elevation data layers.

Generating "synthetic" data can provide a geographically correct representation of the natural and man-made environment (land use, vegetation, man-made features) alleviating a reliance on real imagery or cultural data. It can be generated very rapidly, often procedurally, by combining texture or shader libraries, templates and foundation data such as land use.

As data sources and types become more diverse and require more processing expertise, offloading the data preparation task outside of the SE pipeline to a data provider with specialist data services is a common option. It offers the benefits of leveraging specialist expertise but may impact the agility and responsiveness of the SE pipeline.

In practise, many SE pipelines incorporate a combination of these approaches, but they only address part of the problem. They fail to provide an end-to-end solution that can deal with increasingly big data inputs and perform efficient exploitation and processing of this input data prior to generating and serving up the simulation ready data to the SE client.

SE Generation Tomorrow

This is not a unique Synthetic Environment data problem. In 2018, Air Force Lt. Gen. John N.T. "Jack" Shanahan described the sheer volume of drone imagery and intelligence, surveillance and reconnaissance, (ISR) data captured by the U.S. military and intelligence community, as "an avalanche of data that we are not capable of fully exploiting,". This ISR data glut is exceeding the capabilities of humans to process and exploit even a fraction of this data in a timely manner. The solution is to incorporate automated processes in the form of Artificial Intelligence and computer vision to provide an enhanced capability to integrate disparate sources of data as the volume and variety of sources grow.

Artificial Intelligence (AI) is being widely applied as a means of dealing with the scale of data, its rate of change and the speed at which it can be processed and exploited. In applications closely related and relevant to SE production, it is enabling integration of satellite, mobile phone, drone and airborne sensors to rapidly create digital maps to reflect the current state of roads, buildings, forests, waterways and other features on a country and continental scale - capturing every building footprint in Australia, 18.5m buildings across 945,087 km^2 of Tanzania, with close to 90 per cent accuracy and all 170 million buildings in the United States (Ecopia 2019).

More broadly, the emergence and proliferation of IoT devices, mobile platforms, drones and autonomous vehicles are driving the demand for very fast, real-time AI based data processing. These "edge" computing systems are characterised by the need to perform AI-based speech recognition, object detection, full-motion video analysis, facial recognition, human pose detection with swift response times on platforms with limited computational resources and power. Autonomous driving and augmented reality (AR) require real-time awareness of and interaction with the environment and cannot tolerate the latency of cloud-based processing. By running AI processes at the Edge, data is created and analysed on the device, reducing the need for long-distance client-server communication, minimizing network bandwidth and latency issues, and processing data with the device in milliseconds. What may be considered fast or "real-time" is use case dependent, but in the context of SE, processes running at 30Hz/FPS would be at the

lower end of the acceptable range. By comparison, autonomous vehicle vision systems require similar performance and are conducting image processing based on AI computer vision at rates of 20/30FPS.

Applying this level of fast or real-time AI performance to SE related data processing tasks offers the potential to provide an end-to-end data to SE pipeline with the capacity to rapidly respond to new and changing data inputs. This will require designing AI solutions that achieve suitably accurate data processing at acceptable rates of run-time performance. The following sections will examine issues relating to the training and run-time performance of AI solutions, the benchmark performance of existing example AI data processes relevant to the SE pipeline and the potential for deploying these into an SE pipeline.

ARTIFICAL INTELLIGENCE

Many definitions of Artificial Intelligence (AI) exist, but in essence it refers to the capability of a machine to imitate intelligent human behavior. AI has been subject to periods of advancement and stagnation, constrained by lack of available data and computing power. From the 1990's, Central Processing Unit (CPU) performance improvements and web driven data proliferation accelerated AI development and gave rise to Machine Learning. Machine Learning is a sub-category of AI that learns by utilizing algorithms to parse data, learn from that data, and then apply what they've learned to make informed decisions (Zendesk 2017).

In the last decade deep learning has emerged as a subset of Machine Learning. Deep Learning employs Artificial Neural Networks that are computational models inspired by the human brain. A Deep Neural Network (DNN) (Figure 2) has numerous processors operating in parallel, an input layer, several hidden layers, and an output layer. Each layer is composed of many highly interconnected nodes with weights that adjust the signal between layers.

Developing then applying a DNN to a specific task, and if necessary, to perform this task very quickly, in real-time, is the product of a two-stage process – training followed by inference (Figure 3).

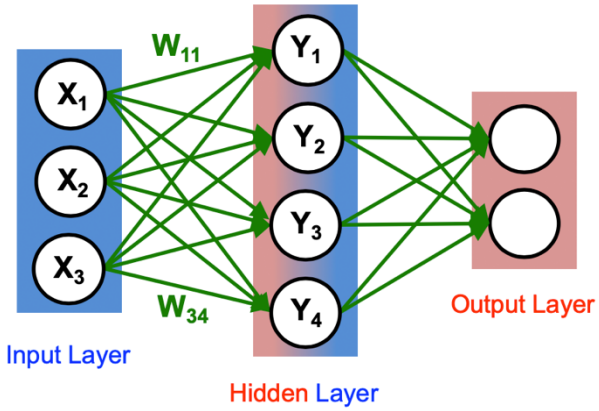

Figure 2. Neural Network Architecture

DEEP LEARNING TRAINING AND INFERENCE

Firstly, models must be trained. Whatever their specific task, recognizing faces or processing speech and natural language, deep learning models require training to learn how to perform their particular job. Secondly, once trained, a neural network is deployed into the digital world as an optimized application that applies this learning to its task by

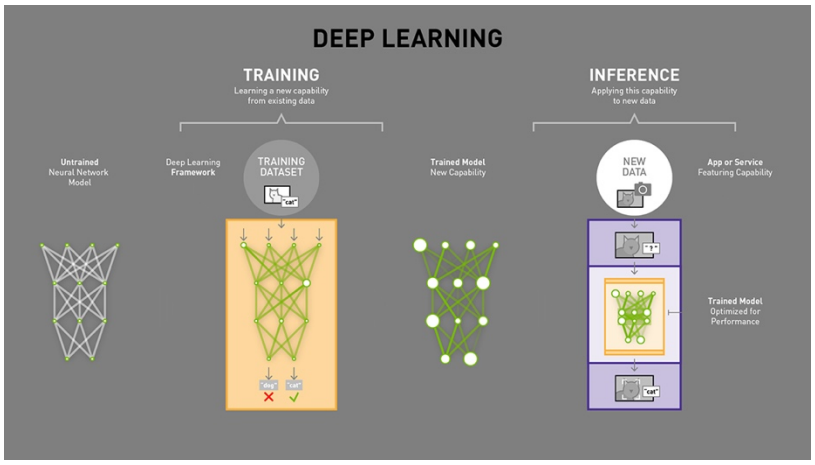

Figure 3. Deep Learning Training and Inference

"inferring" things about the new data it encounters based on its training - in a process referred to as inference. The training and inference phases have importantly different characteristics.

Training

The primary objective of the training phase is to design and train a model to perform a specific task to an acceptable level of accuracy. At this stage the model is learning the salient features and patterns that best represent the data from a labelled training data set. It involves passing sample data through the model and outputting a prediction, expressed as a percentage, that the model has accomplished its task. This cycle is repeated, adjusting the model weights by a process of forward and backward propagation until predictions meet the required accuracy.

The vector and matrix manipulations performed during this process are computationally expensive. Batching training data helps improve computational efficiency and reduce the number of training cycles. Utilising the parallel processing capabilities of Graphics Processing Units (GPU's) can reduce the time taken per training cycle to just seconds compared to minutes on a CPU (these are indicative figures, dependent upon model size, and batch size). The majority of deep learning training is performed on GPU's or dedicated Tensor processing Units (TPU's). Training is characterised by the need for terabytes of labelled data (1000's of images), access to exaflops of parallelised computational resources (GPU's) and the expertise, time and effort to develop the model and run the training process.

Data and computational requirements can be mitigated by the use of transfer learning. This technique applies knowledge gained while solving one problem to solve a different but related problem by leveraging and adapting pre-trained models. An existing model, trained on a very large training dataset can be adapted to a new problem, and re-trained with a much smaller dataset in less time. Algorithmic efficiency improvements have also been reducing the amount of compute needed to train a neural net by a factor of 2 every 16 months since 2012, and it now takes 44x less compute to train a neural network to the level of AlexNet (OPenAI, 2020).

The primary performance metric of training is throughput - the volume of training data per second that can be pushed in batches through network layers – and the time required to train the model to achieve the required accuracy.

Inference

In contrast, inference is focused on deployment of the trained model into a production environment where it may receive a continuous stream of input data and run inference in real time, generating output directly as the end result or as input into other downstream processes or systems.

Fast or real-time inference performance requires specific attention to the efficiencies of the hardware and software aspects of inference, and as illustrated in Figure 4, these are not necessarily the same as those for the training phase.

The metric of performance for inference is latency – the time it takes to push an item through the network. The practical application of DNNs to speech recognition, driverless cars or search engines is only made possible by very fast deep learning inference measured in tens of milliseconds.

Much of the DNN research activity is concentrated on developing models that achieve high levels of accuracy. It does not necessarily prioritise inference performance, and many research papers do not even include details of how fast the model performs. The practical challenges of deploying DNNs across a wide range of applications and platforms has led to co-designing DNN inference software and hardware architectures to deliver optimised DNN models and accelerated hardware solutions that enable efficient inference (Han, 2017).

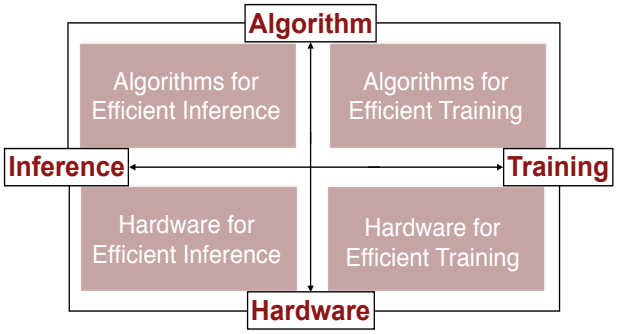

Figure 4. Algorithm & Hardware Efficiency

DNN Software / Algorithms

A property of Neural Networks is that the larger they become, the better they seem to work (LeCun 2019) (i.e. greater accuracy, lower error). Larger means more depth (more layers) and width (more nodes and weights in each layer), resulting in more operations (floating-point operations (FLOP's) or parameter size) that are computationally and memory intensive. Figure 5 illustrates the comparative growth of two image and speech recognition models over the course of just 3 years and 1 year. (Han, 2017).

More weights demand more storage and memory bandwidth. For example, the AlexNet Caffemodel is over 200MB, and the VGG-16 Caffemodel is over 500MB, requiring 15.8 GFLOPs of compute per image (Han, Mao, Dally, 2016). These model sizes impede model deployment and run-time performance, particularly on mobile and edge devices limited to far less compute power and storage.

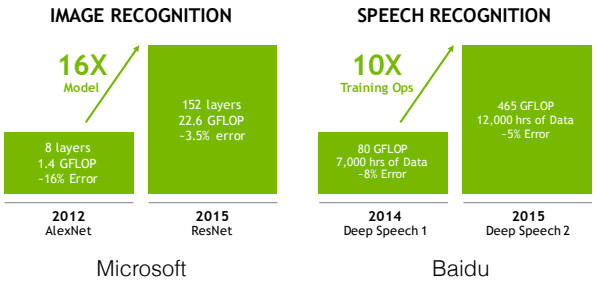

Figure 5. Increasing Model Size

If these models can be compressed to reduce their storage and memory footprint, deployment on lower performance devices becomes feasible and inference speed will improve.

Model Compression

Application of deep compression techniques (Han, Mao, Dally, 2016) combining model pruning, trained quantization, weight sharing, and Huffman coding can reduce the storage requirement of neural networks by 35× to 49× without affecting their accuracy.

Pruning

Model pruning (Figure 6) aims to reduce the size of the model by removing parameters that are not essential to its performance. Pruning takes advantage of redundancy and the sparsity of parameters that do not significantly contribute to model performance.

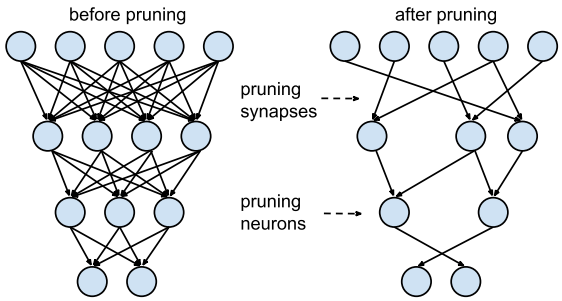

Figure 6. Model Pruning

Trained quantization and weight sharing

Quantization compresses the original network by reducing the number of bits used to represent the weights. Applied to a pruned AlexNet quantization reduces the convolutional layers from 32 to 8 bits without any loss of accuracy (Han, Mao, Dally, 2016).

Following pruning and quantization, the model is retrained to fine tune the remaining connections before applying the Huffman Coding to the quantized weights to provide further lossless data compression.

Performance Improvement

This deep compression method (Figure 7.) compressed storage required by AlexNet by 35×, (240MB to 6.9MB), without loss of accuracy and reduced the size of VGG-16 by 49× (552MB to 11.3MB) again with no loss of accuracy. Benchmarked on CPU, GPU and mobile GPU, the compressed network provided a 3× to 4× layerwise speedup. (Han, Mao, Dally, 2016).

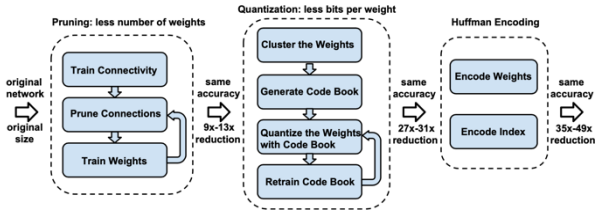

Figure 7. Model Deep Compression

Optimisation

The alternative to compressing and optimising existing models is to design an optimised model from the outset. Models such as Squeezenet, are designed with a smaller Convolutional Neural Network (CNN) architecture containing less parameters whilst retaining and matching the accuracy of larger models. Squeezenet can achieve AlexNet level accuracy with 50x fewer parameters and a compressed model size of less than 0.5 MB (a 510x decrease over AlexNet) (Iandola et al. 2016).

Energy consumption

Beside latency, energy consumption and power have become a major issue for DNN inference. This is particularly true for power constrained mobile, edge and other lighter devices, but even in datacentres, power is a consideration in total cost of operation (Lecun, 2019). Running large neural networks require a lot of memory bandwidth to fetch the weights and a lot of computation to perform dot product calculations. Memory access is the main driver of energy consumption. Large networks that are too big for on-chip storage will require DRAM accesses. Running a 1 billion connection neural network, for example, at 20fps would require 12.8W just for DRAM access - exceeding the power available on a typical mobile device (Han, Mao, Dally, 2016).

Inference Hardware

DNN training and inference are computationally demanding but have different priorities. Training needs high throughput, the massive parallelism of GPU's and efficient processing of large batches of training data. Inference involves operations that can be parallelized, but prioritises efficient memory access, dataflow and fast end-to-end response times in low latency architectures. The inference hardware landscape is complex with GPU, CPU, Field Programmable Gate Arrays (FPGA) and Application-Specific Integrated Circuit (ASIC) architectures all being deployed to target specific use-case applications (Figure 8) (McKinsey 2020).

Nvidia Volta with Tensor Core GPU is focused on inference performance. The CUDA-X AI library and TensorRT SDK provide inference optimisations including support for INT8 and FP16 operations. The Jetson System-on-Module (SOM) devices provide a solution optimised for edge applications.

CPUs offer lower latency but lower parallelisation than GPU's. They are widely deployed for inference in datacentres and currently dominate on mobile devices. The Intel Xeon Scalable processors are targeting inference optimisation incorporating DLBoost INT8 support, Vector Neural Network Instruction and OpenVINO toolkit.

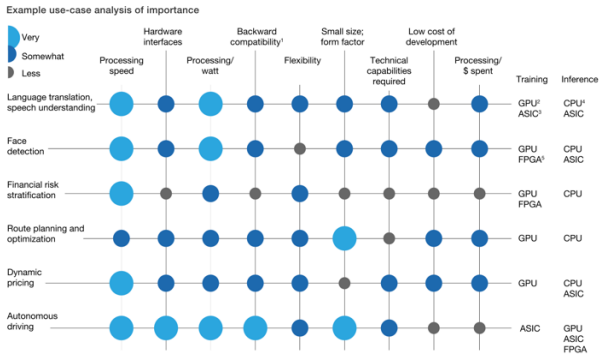

Figure 8. AI Hardware Use Cases

FPGAs provide a flexible hardware configuration, enabling customisation of the inference architecture to a particular application and often provide better performance per watt of power consumption than GPUs. FPGA's are evolving fast, improving their traditionally weaker floating-point performance compared to GPU's. Their customisability is suited to the low precision data types and sparse models now being introduced by optimised DNN's.

ASICS are receiving considerable industrial investment and research in the belief that a dedicated chip design can yield superior performance for one type of computational workload. Google's Tensor Processing Unit (TPU) provides inference specific design optimisations of the major mathematical operations required for DNN inference. and often delivers 15x to 30x faster inference than CPU or GPU (Wang, 2017).

Inference is being run in many places - datacentres, the cloud, on high-performance systems and on edge devices, across a wide range of applications. Datacentre inference

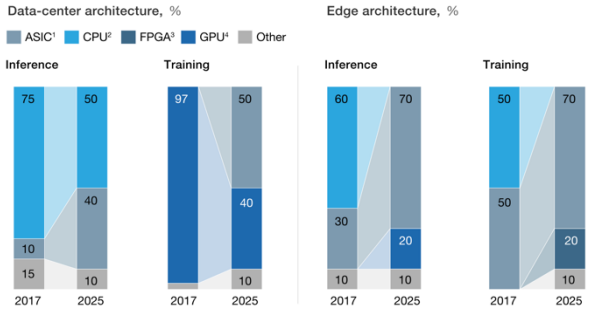

Figure 9. AI Hardware Market Share

is currently dominated by CPU's (75%) (Figure 9 McKinsey 2020). Today Facebook datacentres employ CPU's and perform 3×10^{14} predictions per day, ranging from "simpler" newsfeed ranking to more computationally intense image and video understanding, processing each of the 2 to 3 billion photos uploaded everyday within 2 seconds. (LeCun, 2019). But by 2025, CPU share is predicted to reduce as ASIC's grow to capture 40% of this market. CPUs account for 60% of the current Edge inference market but are predicted to be entirely replaced by ASICs (70%) and GPUs (20%) by 2025.

These trends reflect the expectation that specific AI acceleration hardware will yield better inference performance, particularly at the edge where power consumption, latency performance, memory bandwidth and size are critical. The Efficient Inference Engine (EIE) (Han et. Al. 2016) is an example of the future of bespoke inference acceleration hardware and software co-design. EIE is an energy efficient ASIC accelerator optimised to operate on compressed DNNs on mobile devices, offline, in real-time with low power, delivering a 189x speedup in performance over a CPU.

APPLICATION TO SYNTHETIC ENVIRONMENTS

Applying Deep Learning Inference to the SE Pipeline

Deep Learning solutions can be applied to most of the tasks throughout the entire SE pipeline, from data search and discovery to 3D model generation (Long, 2019). The following examples will focus on the geospatial data processing and preparation activities, and the possibilities of inserting a fast or real-time AI processing capability into this phase of the SE pipeline (Figure 10).

Synthetic environments are now commonly composed of high-fidelity elevation, imagery, point cloud and cultural feature layers, semantic data, topology, material and thermal properties. The features, associated semantics and attributes represented in these supporting data layers are extracted and derived from elevation, imagery, point cloud and similar data sources. Over the last decade, Deep Learning has transformed the field of computer vision and has been widely applied to data exploitation in digital mapping, urban planning, disaster and emergency management, earth science and remote sensing. Computer vision employs Convolutional Neural Networks (CNN) for the tasks of image classification, object detection, and semantic segmentation.

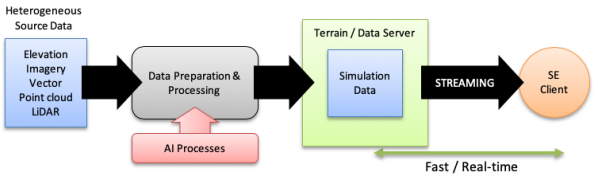

Figure 10. AI processes in the SE Pipeline

Classification assigns a whole image to a class depending on what is in it (Figure 11). Object detection refers to the task of detecting the presence of particular classes of object in an image, localising them with a bounding box and labelling the bounding box. Semantic segmentation assigns a class to each and every pixel in an image based on which image object or region it belongs to. It detects and localizes objects in the image, and also outputs their exact areas and boundaries.

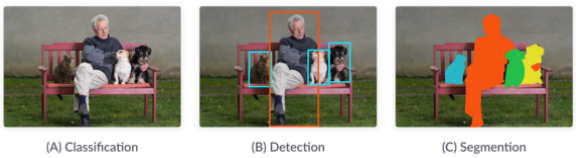

Figure 11. DNN Computer Vision

Deep Learning Performance

Semantic Segmentation is widely used in autonomous driving, industrial inspection, remote sensing and medical imaging analysis and many other fields to help determine relations between objects, as well as the context of objects in an image. It is the primary method used to extract features relevant to SE, such as building footprints, roads and land cover. Semantic segmentation of these features will be used in the following examples to illustrate the level of inference performance achievable with DNN computer vision.

Building Footprint Extraction

Figure 12 illustrates the building footprints extracted for downtown Philadelphia in the course of performing large scale building extraction across the United States (Yang et al. 2018). Using RGB and infra-red bands from 1m resolution aerial images from the USDA National Agriculture Imagery Program (NAIP), the CNN achieved an average processing time of less than one minute for an area of ~56km^2. Using a hybrid fully convolutional network (FCN) (Schuegraf, Bittner, 2019) extracted footprints from a 1.25kmx0.625km area of multi-resolution WorldView 2 PAN, RGB and DSM Munich data in 0.08s.

Figure 12. Building Footprint Extraction

Road Network Extraction

Automated road network extraction from remote sensing imagery can be achieved in similar timeframes. The City-scale Road Extraction from Satellite Imagery (CRESI) (Van Etten, 2019) model combined the SpaceNet 3 algorithm, a ResNet34 encoder and a U-Net based decoder with a process that can infer road networks from input images of arbitrary size. CRESI was able to achieve an inference speed of 160 km^2 / hour. Figure 13 illustrates the CRESI generated roads for Shanghai. CRESIv2 (Van Etten, 2020) has improved this speed to 280km^2/ hour. In comparison, using a DenseUNet model, (Xin, Zhang, Zhang, Fang, 2019) achieved road extraction at speeds of 0.3 – 0.4ms per 2.25km^2.

Figure 13. Road Network Extraction

Land Cover

Land cover and land use classification apply semantic segmentation to identifying different feature classes, which may include buildings, water, forests, marsh, fields, impervious surfaces or vegetation. Extraction can be performed on a variety of remote sensing data including multi-spectral as well as SAR imagery. In a study to evaluate seven state-of-the-art semantic segmentations models (U-Net, DeepLabV3+, PSPNet, BiSeNet, SegNet, FC-DenseNet, and FRRN-B) land cover classe extraction from ESA Sentinel-1 C- band SAR images achieved inference speeds between 0.02 and 0.19 seconds for a 100km^2 area (Figure 14) (Šćepanović, Antropov, Laurila, Ignatenko & Praks, 2019).

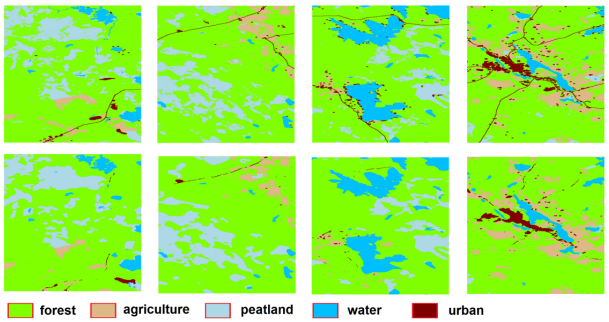

Figure 14. Land Cover Extraction

Point Cloud

Photogrammetric data collection methods collect point cloud data of natural and man-made terrain and cultural features and can be processed into a form usable for SE construction using CNN based segmentation techniques. The Semantic Terrain Point Labelling System Plus - STPLS+ (Chen et al. 2019) utilises a U-Net CNN to segment the point cloud into ground, man-made objects, and vegetation. (Figure 15) This work is focused on the experimentation and practical application of this technique to the SE need to classify and recognize objects and their associated properties (location, materials) in

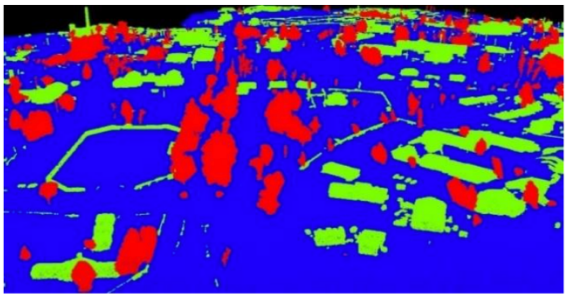

Figure 15. Point Cloud Extraction

support of visual rendering and user interactions with the environment. This approach achieved processing times of 2 hours/km^2 on the Muscatatuck Urban Training Center (MUTC) (an area of approximately 4km^2).

Analysis

These inference performance times are indicative of the inference speed achievable with these techniques. The development of these models prioritised training to an acceptable level of accuracy rather than inference performance. Inference speeds were achieved testing models on a subset of training data, using relatively modest hardware, with no specific optimisation for inference performance. Comparison and benchmarking of DL model performance is not straightforward. The numerous possible combinations of inference applications, data sets, models, machine-learning frameworks, tool sets, libraries, systems, and platforms complicate systematic and reproducible benchmarking. (Torelli, Bangale, 2019).

A similar complexity is reflected in the examples, making reliable, like-for-like comparison of inference speeds difficult. Table 1 summarises key characteristics and differences relating to the data sets used in the examples, extracted from the research papers. A rudimentary standardisation of the time taken by each model to process 1 km^2 has been derived from the inference speeds (yellow column). Although fast, these models are not achieving the 20/30FPS real-time threshold. There is a significant speed difference between the CRESI and DenseUNet model (Xin, Zhang, Zhang, Fang, 2019). The lower resolution sample data of the DenseUNet may be a factor, but this cannot be determined without further evaluation.

Table 1. Inference Performance Summary

Reference	Features	Test Data	Res	Bands	Sample size k pixels	Reported Speed (seconds)	Reported Sample Area sq km	Seconds per sq km	Hardware
Yang et al.	Buildings	NAIP	1m	RGB+IR	6 x 7	60	56	**1.07**	NVIDIA Tesla K80 GPU
Schuegraf, Bittner,	Buildings	World View-2	0.46m	RGB, PAN,DSM	0.32 x 0.32	0.08	0.0256	**3.125**	NVIDIA GeForce Titan X
Van Etten	Roads (CRESI)	worldview 3	0.3m	RGB	1.3 x 1.3	3600	160	**22.5**	NVIDIA Titan X GPU
Van Etten	Roads (CRESIv2)	worldview 3	0.3m	RGB	1.3 x 1.3	3600	280	**12.85**	NVIDIA Titan X GPU
Xin, Zhang, Zhang, Fang	Roads	massachusets data	1m	RGB	1.5 x 1.5	0.472	2.25	**0.2**	GTX1080Ti
Šćepanović et al.	Land Cover	ESA Sentinel-1 C- band SAR images	20m	SAR	0.512 x 0.512	0.2	100	**0.002**	NVIDIA GeForce GTX 1080
Chen et al.	Point Cloud	Photogrammtric point cloud				7200	1	**7200**	Unknown

Comparison of inference performance does not consider the potentially different qualitative criteria of the model outputs – for example, are all models aiming to capture all buildings of any size? – or the difficulty in accurately classifying all pixels under a wide range of conditions. For buildings alone, the sheer geographic diversity of building style, size, shape and density make it difficult to design a universal model that will achieve accuracy regardless of location. Diversity of input data is a further complication – quality, resolution, sensor type, wavebands, area coverage and image size all impact model accuracy and performance. A generic, robust and scalable solution is not yet available (Yang et al. 2018).

Nevertheless, the inference performance of these models is encouraging. Although the examples do not achieve real time rates, the current performance is good enough to provide speedup, automation and extra agility into the pipeline. For example, on a 4 GPU cluster, CRESIv2 could map the entire road network across the 9100 km^2 area of Puerto Rico in ~ 8 hours (Van Etten, 2020) – not real-time, but better than the 2 months required by human labellers. It is also encouraging that these speeds have been achieved without focusing on inference speed and suggests there is scope to improve and optimise performance. However, with so many factors influencing inference speed for each specific DNN architecture and use case, only implementation, test and evaluation can reliably determine the impact of optimising the inference hardware and software architecture.

Nonetheless, there are some general conclusions that can be drawn. Improving the inference hardware is likely to accelerate inference speed and deliver a quick win improvement. But achieving anything close to real-time speeds will require inference to be performed close to the data on a local platform - and begins to move SE related inference toward the paradigm of inference at the Edge. All of the examples were run on relatively modest single GPU hardware. Particularly when running a complex model not specifically designed for fast inference, higher performance hardware hosting multiple GPUs may offer the most effective short-term solution. CRESI improved inference rates from 160 km^2 per hour on a single GPU machine to a minimum of 370 km^2/ hour on a four GPU cluster.

It is virtually impossible to predict the specific level of improvements that can be achieved by optimising any trained model to increase inference speed. However, there are some established general principles. Accuracy and inference time are in a hyperbolic relationship, where a little increment in accuracy costs a lot of computational time – optimisation will become a trade-off between speed and retaining acceptable levels of accuracy (Canziani, Paszke , Culurciello, 2016). There is a linear relationship between the number of model operations and inference speed – reducing the number of operations will improve inference time. Many models like VGG or AlexNet are oversized, under utilise their large number of parameters and do not maximise their potential learning ability. Applying model compression (Han et al., 2015) can reduce network file size by up to 50×.

Alternatively, these principles can be used to design more efficient DNN architecture with an optimum balance between accuracy and inference speed from the outset. There is a growing number of models specifically designed for efficiency including ICNet, BiSeNetV2, Enet, Segnet, DLC, ERFNet, ESPNet, GUN and DFAN. The Image Cascade Network (ICNet) (Zhao, Qi, Shen, Shi, Jia, 2018) can achieve real-time inference and semantic segmentation on high-resolution images on a single Maxwell TitanX GPU card. The framework progressively refines segmentation prediction on a cascade of image inputs (i.e., low, medium and high-resolution images). Compared to the PSPNet50 baseline, ICNet achieved a 5x speedup of inference time, 5x reduction in memory consumption, and can run at high resolution (1024×2048) speeds of 30 fps (cityscapes dataset). The Bilateral Segmentation Network (BiSeNetV2) framework is a generic architecture, which achieved inference speeds of 156 FPS on the Cityscapes dataset running on an NVIDIA GeForce 1080Ti card (Yu et al. 2020).

The inference speeds being achieved by the examples in this paper and the improving performance of other optimised models demonstrate that the capability and the potential to apply AI to relevant, fast data processing exists now. Integrating AI data processing into the pipeline (Figure 16), potentially as on-demand service on the terrain or data server would enable a true end-to-end capability, able to accept raw data, process and prepare it at real-time rates before passing it to the streaming / rendering / publishing phase. However, the examples are not achieving these real-time rates today, and it will almost certainly require development of an optimised software (DNN architecture) and hardware solution. Although there is rapid innovation and change in real-time deployment of AI, even the biggest players (Facebook) recognise that co-designing DL inference hardware for current and future DL models is an important but challenging problem (Park et al. 2018).

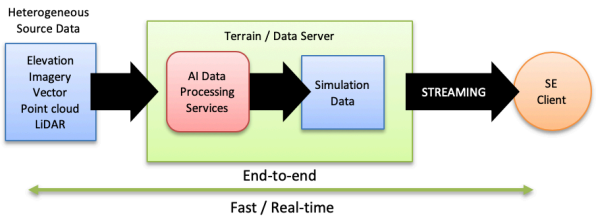

Figure 16 Integrated AI

Apart from speed, incorporating AI into the pipeline introduce other considerations. DNNs have been described as "black boxes" with no way to determine how the algorithm came to its decision. Understanding and being confident in the results of one DNN in one system is challenging. A networked scenario may have multiple and different DNN's, ostensibly performing the same task but potentially producing different results even if they act on the same input data. This has potentially negative interoperability implications. Applying AI explainability methods and techniques can help address these issues.

At the moment, it is usually necessary to design the DNN according to specific tasks to obtain the best performance. Even performing the same class of task, extracting a particular feature type, may not be possible with exactly the same model without at least retraining it to account for significant differences in the data it encounters. A future pipeline must be capable of dealing with a wide range of data and will require models that can autonomously learn and adapt to new data and be capable of continual learning.

CONCLUSIONS

DNN are fast relative to alternative methods, but many models prioritise accuracy over inference speed. Selecting a DNN to deploy to a specific task is complicated by the complex combinations of data sets, models, frameworks and lack of systematic benchmarking. The example applications selected for their relevance to SE have developed effective DNN's designed for accuracy, and although not achieving real-time rates, offer significant speedup to the SE pipeline and are an indicator of the capability and benefits to be gained from the broader utilisation of DNN. The significant

innovation and rate of evolution of DNN optimisation is being driven by the growing needs for efficient, fast inference particularly for mobile and edge devices. These applications are delivering real-time inference speeds now and show what is possible if applied to SEs. But this is likely to require similar software and hardware co-design specifically aimed at the SE pipeline to achieve the required performance for SE specific tasks. Although this topic is the subject of much innovation and research, this is a challenging goal. But it offers the potential to significantly shift the SE construction process toward a solution that is able to deal with big data with far greater speed and agility than is currently possible, with a similarly positive impact on the speed and quality of the SE.

REFERENCES

Canziani A, Paszke A, Culurciello E (2016). An Analysis of Deep Neural Network Models for Practical Applications. Retrieved From: https://arxiv.org/abs/1605.07678v4

Chen M, Feng A, McCullough K, Prasad, McAlinden P, Soibelman L, Enloe M (2019). Fully Automated Photogrammetric Data Segmentation & Object Information Extraction Approach for Creating Simulation Terrain (IITSEC 2019).

Ecopia (2019). Partnership Profile: Maxar Technologies. Retrieved From: https://medium.com/@ecopia.ai/partnership-profile-maxar-technologies-dd9e7e727522

Han S, Mao H, Dally W (2016). Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. Retrieved From: https://arxiv.org/abs/1510.00149v5

Han S. (2017). Efficient Methods and Hardware for Deep Learning Lecture Retrieved from: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture15.pdf

Han S, Liu X, Mao H, Pu J, Pedram A, Horowitz M, Dally W (2016). EIE: Efficient Inference Engine on Compressed Deep Neural Network. Retrieved from: https://arxiv.org/abs/1602.01528v2

Iandola F, Han S, Moskewicz M, Ashraf K, Dally W, Keutzer K (2016) Squeezenet: Alexnet-Level Accuracy With 50x Fewer Parameters And <0.5mb Model Size Retrieved From: https://arxiv.org/abs/1602.07360v4

LeCun Y (2019). 1.1 Deep Learning Hardware: Past, Present, and Future. *2019 IEEE International Solid- State Circuits Conference* - (ISSCC). doi: 10.1109/ISSCC.2019.8662396

Lexie Yang H, Yuan J, Lunga D, Laverdiere M, Rose A, Bhadur B (2018). Building Extraction at Scale using Convolutional Neural Network: Mapping of the United States. Retrieved From: https://arxiv.org/abs/1805.08946v1

Long G (2019). Building the World - Could AI build our Synthetic Environments? Paper presented at the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC 2019)

McKinsey & Company (2020). Artificial-intelligence hardware: New opportunities for semiconductor companies Retrieved from: https://www.mckinsey.com/industries/semiconductors/our-insights/artificial-intelligence-hardware-new-opportunities-for-semiconductor-companies#

OPenAI, 2020. Retrieved from https://openai.com/blog/ai-and-efficiency/

Park J, Naumov M, Basu P, Deng S, Kalaiah A, Khudia D, Law J, Malani P, Malevich A, Nadathur S, Pino J, Schatz M, Sidorov A, Sivakumar V, Tulloch A, Wang X, Wu Y, Yuen H, Diril U, Dzhulgakov D, Hazelwood K, Jia B, Jia Y, Qiao L, Rao V, Rotem N, Yoo S, Smelyanskiy M (2018). Deep Learning Inference in Facebook Data Centers: Characterization, Performance Optimizations and Hardware Implications. Retrieved from: https://arxiv.org/abs/1811.09886

Šćepanović S, Antropov O, Laurila P, Ignatenko V, Praks J (2019). Wide-Area Land Cover Mapping with Sentinel1 Imagery using Deep Learning Semantic Segmentation Models. Retrieved From: https://arxiv.org/abs/1912.05067

Schuegraf P, Bittner K (2019). Automatic Building Footprint Extraction from Multi-Resolution Remote Sensing Images Using a Hybrid FCN Recovered From: https://doi:10.3390/ijgi8040191

Torelli P, Bangale M (2019). Measuring Inference Performance of Machine-Learning Frameworks on Edge- class Devices with the MLMarkTM Benchmark. Retrieved from: https://www.mlmark.org

Van Etten A (2019). City-scale Road Extraction from Satellite Imagery. Retrieved From: https://arxiv.org/abs/1904.09901

Van Etten A (2020). City-Scale Road Extraction from Satellite Imagery v2: Road Speeds and Travel Times Retrieved From: https://arxiv.org/abs/1908.09715v3

Xin J, Zhang X, Zhang Z, Fang W (2019). Road Extraction of High-Resolution Remote Sensing Images Derived from DenseUNet Recovered From: https:// doi:10.3390/rs11212499

Yu C, Gao C, Wang J, Yu G, Shen C, Sang N (2020). BiSeNet V2: Bilateral Network with Guided Aggregation for Real-time Semantic Segmentation. Retrieved From: https://arxiv.org/abs/2004.02147v1

Zendesk (2017) A simple way to understand machine learning vs deep learning https://www.zendesk.com

Zhao H, Qi X, Shen X, Shi J, Jia J (2018). ICNet for Real-Time Semantic Segmentation on High-Resolution Images Retrieved From: https://arxiv.org/abs/1704.08545v2

Wang Y. (2017). Deep Learning in Real Time - Inference Acceleration and Continuous Training. Retrieved from: https://medium.com/syncedreview/deep-learning-in-real-time-inference-acceleration-and-continuous-training-17dac9438b0b