NAVIGANT RESEARCH

WHITE PAPER

The Data Driving Automated Vehicles

Where It Comes from and Why Quality Matters

Published 3Q 2018

Commissioned by Mighty AI

Sam Abuelsamid

Senior Research Analyst

John Gartner

Director

INTRODUCTION

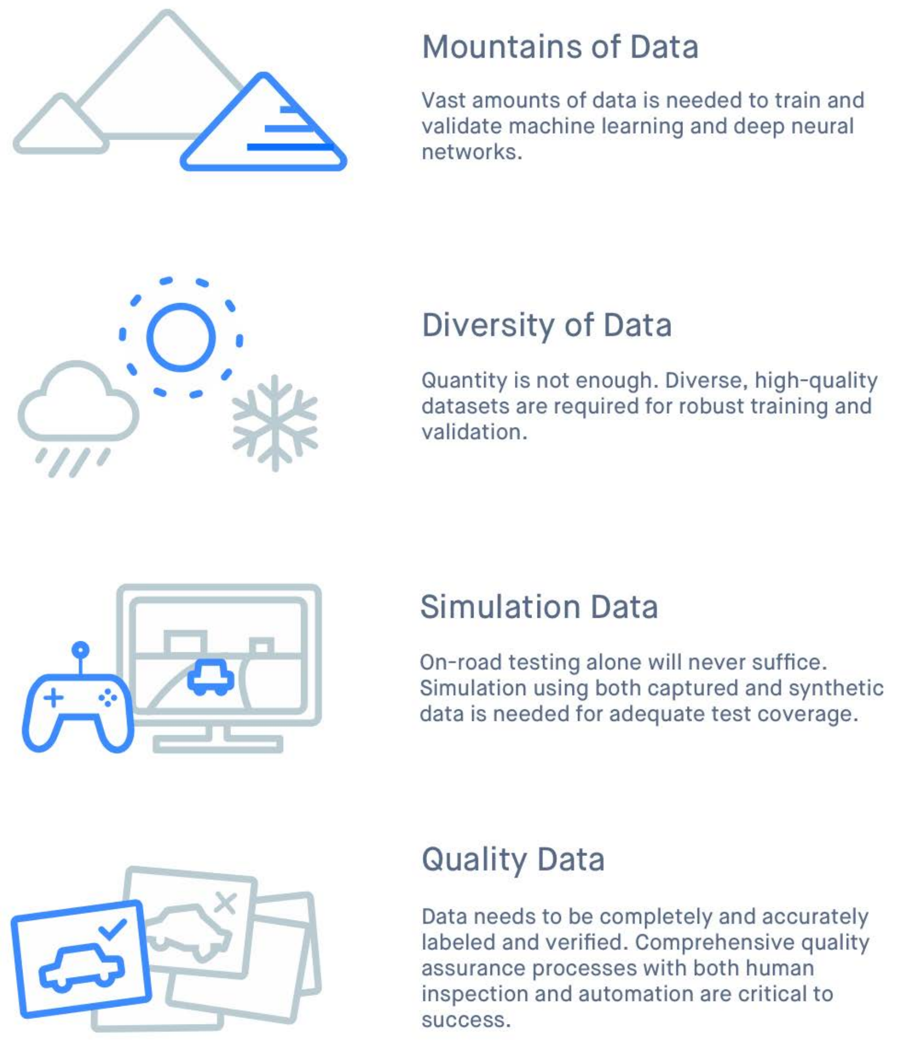
The nature of software architecture and development has shifted dramatically in the past decade, enabled in part by the continuing evolution of highly parallelized microprocessors. Machine learning (ML) and neural networks have become practical to use, allowing engineers to apply these techniques to solve a wide range of esoteric problems. Among these is object recognition and classification which is crucial to enabling everything from manufacturing to security systems to automated driving. In the realm of automated driving, accuracy in object classification and tracking is particularly important because it will influence life or death decisions made by vehicle control systems.

ML and deep neural networks require large quantities of data for both training and validation. A 2012 study by Google Research and Carnegie Mellon University found a logarithmic improvement in machine vision performance as the size of the datasets was increased from 1 million images to 375 million images. Automated vehicles being tested today may be equipped with 40 or more sensors and with the capacity to capture 4 TB of data or more per day.

However, throwing masses of raw data at the problem does not make magic happen. The data used has to be accurately labeled and annotated so that the models understand what is being detected. For both training and testing purposes, a sufficiently large dataset is needed to provide complete coverage of both positive and negative results. Datasets need to include not only the common, everyday scenarios the system will encounter but also more rare or challenging targets such as a large pothole filled with water on a rainy night or a load of eels dumped from a truck onto a roadway as happened in Oregon in July 2017. Finally, the validation dataset needs to be distinct from the training set to ensure that the models actually work correctly.

This report examines the use cases, issues, and requirements for data used for automated driving and related mobility applications.

Key Takeaways



Mountains of Data

Vast amounts of data is needed to train and validate machine learning and deep neural networks.

Diversity of Data

Quantity is not enough. Diverse, high-quality datasets are required for robust training and validation.

Simulation Data

On-road testing alone will never suffice. Simulation using both captured and synthetic data is needed for adequate test coverage.

Quality Data

Data needs to be completely and accurately labeled and verified. Comprehensive quality assurance processes with both human inspection and automation are critical to success.

Source: Mighty AI

THE NEED FOR DATA

Whether it is a human or a machine, the process of learning requires data and rules. Rules enable criteria to be defined for decision-making, the data provides the examples used to calibrate the decision rules. Accurate training or calibration of a system—whether it is organic or digital—requires the processing of a certain amount of data to have a sufficiently high degree of confidence in the ultimate decision-making. A child may only need to see a book once or twice to grasp its pictures and understand its contents. Conversely, a deep neural network (DNN) may need see hundreds of thousands or even millions of images of cyclists from various angles and in different conditions to reliably understand what that is. Processing more data and evaluating the results of the subsequent decisions can help refine the model over time to either make it more accurate or identify a need for additional criteria.

Where Is the Data Used?

Teaching the Car to Drive Itself

The classical approach to developing electronic control systems involves using sensors to directly detect the behavioral traits of the vehicle and driver. That knowledge is used to manage actuators that effect that behavior. For example, anti-lock brake systems use wheel speed sensors to measure each wheel's rotational speed and the brake pedal switch to determine if the driver was trying to slow down. From the wheel speeds, other data points such as a vehicle reference speed, wheel acceleration, and slip could be synthesized. These and other data points were then fed into a deterministic rule set that managed how much braking pressure should be applied to each wheel in order to provide the optimum balance of vehicle stability and stopping distance.

The limited number of data points and resolution of that data make this approach both manageable and usable. Effectively all the system was trying to measure was available grip at the tire/road interface and how to optimize the use of that tractive force. This is easily characterized from wheel speed data. Combined with an understanding of the kinematic behavior of the vehicle systems and actuators, reliable control of the brakes is achievable with relatively low powered processing.

As engineers began trying to replicate visual processing for automated driving, it became clear that this rules-based approach with calibrations was not up to the task. While characterizing the behavior of the wheels and how that relates to overall vehicle behavior is straightforward, classifying the objects detected by cameras, lidar, and radar and understanding the scenarios they participate in is a vastly more complex problem. The human brain is capable of processing incredibly complex data from the senses and detecting subtle nuances. Humans can distinguish the difference between a child, a dog, a deer, another vehicle, or a plastic bag blowing across the road and crucially can read subtle cues to predict what those objects may be do next, all areas that software has traditionally struggled with.

The development of machine learning (ML) approaches based on DNNs is widely considered to be the key to the successful development of highly automated driving. While traditional deterministic algorithms gave the engineers virtual knobs and switches to fiddle with in order to optimize the performance, ML works in a fundamentally different way. A DNN consists of multi-layered models that are trained by feeding in data of the type that is expected to be processed. As a DNN model is fed more data, it creates internal connections and essentially calibrates itself, much as the human brain does when synapse connections are created. Engineers typically have some ability to tune models but much of the effort happens automatically—whereas for humans, initial guidance about desirable or undesirable characteristics is necessary. Parents indicate to children not to touch a hot stove and when a hand senses the heat they know not to touch.

Similarly, DNNs need guidance. That comes in the form of data that has been meticulously labeled and annotated. As the data is read, the model comes to understand the difference between a person and a cat, between a box and a car. Even in cases where more classical software engineering methods are used, data guides the engineers that are refining the rule sets, algorithms and calibration.

On the Road

From the earliest days of the automobile in the late-19th century, development of the mechanical systems was done by testing hardware, first on the bench and then in complete vehicles on the road. For purposes of this discussion, road testing includes evaluation both on closed test tracks and on public roads.

Road testing remains a crucial component of the vehicle development process, despite the fact that there are vehicles under development that someday will not need a driver or even a human supervisor. Between weather, road conditions, and the unpredictable behavior of other road users, valuable data can be gathered. This includes the raw data of what the sensors are seeing, logs of what the software is doing, and ultimately the vehicle performance data. The logs and performance data enable engineers to evaluate the system performance and debug issues.

The variability and unpredictability of what can happen on the track or the road can be captured through the sensor data. Many on-road scenarios are typical of day-to-day driving and are relatively straightforward to reproduce. However, there are far more situations that occur so infrequently that they are often difficult or even impossible to reproduce on the road. While each of these so-called edge cases typically do not happen often, they are often the most challenging to deal with for a human and more likely to result in a crash. Capturing the data is crucial to understanding the system behavior and developing it further to improve its robustness. Edge cases are also difficult for engineers to anticipate. For example, a July 2017 crash in Oregon that dumped thousands of pounds of slime eels on the road is not a situation that engineers would be likely to anticipate and build simulation models for. It also would not be readily reproducible with hardware testing. However, having data captured from vehicles in the area or synthetic data would allow repeated testing of potential solutions in simulation.

A highly automated vehicle can generate 4 TB or more of raw data per day. This is far too much to transmit in real-time while driving, so it is typically captured on high speed storage drives during testing. At the end of the day, technicians will typically plug in each car as it comes back to the garage and ingest that data into network storage, often both locally in the test facility and in the cloud.

In normal deployment, the majority of this data will be short-lived, used only for real-time control of the automated systems and then discarded. However, when anomalies are detected, relevant data can be captured and transmitted for use in improving the system. During development testing, an engineer or technician monitoring the vehicle's behavior can flag incidents for later offline review. Tesla has been using a shadow mode during the ongoing development of its Autopilot system. Prototype algorithms run in the background while either a human driver or current production software is in control. When differences in decisions are made, relevant data can be captured and transmitted back to Tesla for analysis.

In the Simulator

Automated driving technology remains immature while being vastly more complex than any previous electronic control system. While many safety critical control systems have been developed, none, including aircraft, have had to operate autonomously in environments as variable as driving. Automated vehicles must operate and safely coexist with unknowns that range from weather to road conditions to infrastructure to pedestrians and other human-driven vehicles. The spectrum of operational scenarios is virtually limitless, and it is not realistic to be able to test every system change against all expected conditions in the real world.

This is where simulation will prove to be critical. As an example, Waymo has accumulated more than 8 million miles of on-road testing since its development program began in 2009. In 2017 alone, Waymo accumulated the equivalent of 2.7 billion miles in simulation environments and continues to rack up millions of miles more every day. For those miles to have value in validating the efficacy of the system, hundreds of thousands of unique scenarios are required. The different types of sensors in use include cameras, ultrasonic, radar, and multiple types of lidar, they all have different operational characteristics. Cameras in particular are sensitive to lighting conditions, while lidar effectiveness is influenced by the reflectivity of surfaces being detected.

Simulation provides the ability to repeat a nearly infinite number of scenarios using a combination of raw data recorded from vehicles and synthetic data created from scratch or through manipulation of recorded data. Chip maker Nvidia has done extensive work, using its powerful graphics-rendering capabilities to make small adjustments to lighting, surfaces, and even weather conditions that can be fed into simulations that range from pure software in the loop where the hardware itself is simulated before making prototypes to full-blown hardware in the loop.

Powering Services for New Revenue

Automated driving systems hold much of today's attention; however, the potential to use data for mobility extends well beyond replacing humans with virtual drivers. Among the many applications that are already in use are risk models for usage-based insurance and building high definition maps and mobility logistics platforms. As ML continues to mature, the technology is being applied to develop what will hopefully be ever-more sophisticated models. Such models can provide both more accurate risk predictions used for pricing policies for individual drivers and guidance for drivers to hopefully help them become safer.

Telemetry data from vehicles correlated with service history can be used to build models useful for predictive diagnostics. This is especially valuable to fleet operators that rely on avoiding unpredicted downtime. Having accurate projections of when service will be needed can aid fleet availability while at the same time avoiding unnecessary service, helping the bottom line. Even individual consumers can benefit from predictive diagnostics by enabling service to be scheduled at their convenience before a vehicle is stranded. There are also revenue opportunities for OEMs when they can alert customers and then make the connection for a service appointment with provider partners.

Another prime use case for ML in the coming years will be automotive cybersecurity. OEMs have become increasingly attuned to the security concerns around connected vehicles, especially those that include automation capabilities. In response, numerous companies have launched with solutions designed to help prevent malicious attacks on vehicles. Many of these are using ML to improve detection mechanisms. Training is done with data captured in vehicles when any sort of anomalous behavior is detected. The data then is sent back to the vendors data center for analysis and reuse in training updated software.

WHERE DOES THE DATA COME FROM?

There are two sources of data used in product development and validation, raw real-world data captured from real hardware on the bench or in the field, and synthetic data that can either be modified from real data or generated from scratch.

Capturing Raw Data

A fully equipped automated vehicle can have dozens of high resolution sensors installed on it, each capable of capturing multiple gigabytes of data per hour. In total, an automated driving development vehicle may collect 4 TB of data or more per day during on-road testing. Depending on the application, data can be collected in multiple ways.

For the most intensive applications, such as highly automated driving, test vehicles will typically be configured with a network-attached storage system alongside its compute platform. With a high speed network installed in the vehicle, all the raw sensor data will be captured in real-time onto the banks of hard drives. A typical scenario will involve vehicle operators driving prescribed routes for several hours a day, often covering the same roads multiple times in order to log variations in the conditions over time.

At the end of a shift, operators return to a depot and plug their test vehicle into the local network where the data is ingested into a storage area network in the facility. In some cases, removable hard drives are used that are pulled from the vehicle and docked for data ingestion. Upon completion, the data is wiped, leaving room to capture data for the next day. In most cases the data is also uploaded from local storage to a data center or cloud provider where it can be used by teams at other locations or shared to vendors that do data processing, such as labeling and annotation.

Map-making is another example where large quantities of detailed data can be collected by data collection vehicles and then ingested for processing in a similar fashion to automated driving development. As mapping providers build out increasingly detailed maps in the coming years, using a mix of their own fleets and crowd-sourced data will be necessary.

Few data-intensive applications can capture, process, and transmit data from the field. For example, many automakers capture telemetry data from customer cars using built-in cellular telematics connections. This data can be aggregated and analyzed to gain insights about how vehicles are used, and that information can be fed into future product development. It can also be used to provide feedback to customers or fleet managers for predictive diagnostics. Telemetry data can be used to train ML algorithms that can predict likely component or system failures before the failures happen, giving the vehicle owner an opportunity to seek out a repair before being stranded.

In some cases, raw data is processed locally on board the vehicle to detect anomalies or items of interest. For example, manufacturers are beginning to launch vehicles with vision systems that will look for stationary objects or road features as vehicles drive and compare that to an existing map. When new or changed objects are detected, that can be transmitted to a data center in near real-time and then incorporated into high definition maps. This data can be analyzed, integrated into the maps, and then have updates distributed back to the fleet.

Creating Synthetic Data

Long before there was any widespread use of ML, simulation has been used in engineering work. Various forms of software in the loop (SIL) and hardware in the loop simulations have been a part of the normal course of product development for decades. SIL is typically used in the design analysis that occurs early on in a program before prototype hardware has been fabricated. While data recorded from previous generation systems can be used as an input for these simulations, the addition of new functionality or use cases often drives the need for data that does not yet exist.

Scenarios and use cases that may have been observed but not yet captured also need to be tested. This is particularly true as systems become more complex and as they are intended to work in a broader range of environments. Most initial deployments of automated vehicles are expected to be in urban areas with warm climates and good infrastructure such as lane markings and clear signage. Over time, these vehicles will need to operate in suburban and rural areas and in all weather conditions. Another important variable that needs to be accounted for are local customs, such as how people behave at intersections, whether they consistently use signals, and how pedestrians behave.

Synthetic data becomes enormously valuable in this aspect of the development process. Creating input data from scratch or modifying existing data to replicate new use cases enables developers to test new ideas much sooner and with greater efficiency. This is especially true when considering so-called edge cases, those that are challenging to repeat and capture in the real world or that happen relatively rarely.

For Training and Validation

For vision-based systems that process a sequence of two-dimensional images, modern computer-generated imagery (CGI) can produce photorealistic input signals that can be used for training neural networks for image recognition. Generating synthetic data inputs for other sensor types that “see” in three dimensions such as radar and lidar requires different data formats that replicate the output of a range of sensor types that may be used. One of the advantages to such synthetic data sources is that every object in the field of view can be labeled as it is generated, a crucial aspect for any development project.

There are, however, risks to using synthetic data. Neural networks can still be relatively brittle and sensitive to bad data. The results of training a network can be significantly affected or impaired by data that varies even slightly from the ground truth. As good as modern CGI is, it can still include visual artifacts that can negatively influence the ultimate performance and reliability of the network. Even in real-world scenarios, it is possible to spoof an object recognition network. Researchers have demonstrated that strategically applying a few pieces of tape to a traffic sign can cause it to be misclassified. Similar defects in synthetic data can have unexpected performance results.

To achieve proper results, it is essential to understand the physics of the real-world scenarios that are being synthesized. For example, the reflectivity of objects based on the range of different finishes that will be encountered and even the properties of rain-drops or snowflakes as they fall can all affect the results. Taking a video sequence of a car driving down a road on a clear, sunny summer day and manipulating it to appear like a snowfall or a thunderstorm can save a lot of time in development and testing. However, if any of the variables being manipulated do not behave like the real environment, the resulting network may be useless. Whether using raw or synthetic data, validation that it generates the expected results is crucial before putting it to work.

Early Hardware Analysis

In addition to training the ML systems, synthetic data is valuable in designing the hardware that feeds the algorithms. Mathematical models of the performance of the hardware can use synthetic or real data inputs and then pass along further synthesized data to other levels of the system. As with using synthetic data for training and validation, it can also be problematic for early hardware design. Erroneous synthetic data fed into a simulation model of a sensor can lead to design optimizations that create new problems when actual hardware has been fabricated.

Similarly, feeding the outputs of a simulated sensor into a network for training is probably a bad idea. Artifacts generated either by the data fed into the hardware model or the model itself can lead to unpredictable behavior in the networks and the remainder of the stack. Synthetic data is best used for these types of applications for unit testing when the stack has been architected into modules that can be independently analyzed.

Adversarial Examples

To have a comprehensive dataset for neural networks, both positive and negative or adversarial examples are recommended with appropriate labeling. The positive examples help to teach the system what it should be looking for. However, since current DNN models can remain sensitive to subtle changes in the image being examined, it is important to understand the ways in which things can go wrong. Neural networks need to be evaluated and developers should bring in as much so-called bad data as is available to validate the recognition accuracy.

A 2014 Cornell University study, "*Explaining and Harnessing Adversarial Examples,*" by Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy examined how adversarial examples can generate inaccurate results from image recognition systems. Subtle perturbations that shifted pixels in a way not really discernable to the human eye caused a neural network to completely misclassify images. While this is an extreme example that is possible to reproduce in the test environment, such spoofing is much more difficult in the real world.

The positive side of this is that, while a human can readily process what the underlying image was, the fact that software can discern those extremely subtle differences means that it should be possible to train it to reject them. Doing so requires a large dataset that has been vetted carefully to help the system know what something is not in addition to what something is.

Similarly, the positive and negative examples can be used in systems where redundancy and diversity are required for extra safety. Automated driving is a prime example of this. Having two or more sets of discrete algorithms examine a scene in different ways and then compare the results helps increase confidence in the result and provides a way to detect errors. Using networks to detect other road users and empty space in parallel is one way these redundancies work, and this has been done by companies such as Argo AI and Waymo. While one system detects pedestrians, cyclists, and other vehicles, the other is looking for the areas in each frame where there is nothing. If the inverse of the empty space detection does not match the object detection, there is a flaw in the system.

Labeling and Annotating the Data

The single most important phrase when considering the use of data is “garbage in, garbage out.”

Before any data can be used for building, training, or validating any software system, it must first be understood. At best data that is not representative of what is needed for a particular application will give irrelevant results. At worst it will send developers down blind alleys with ML systems that are detecting entirely the wrong results or trying to correct problems that may not exist.

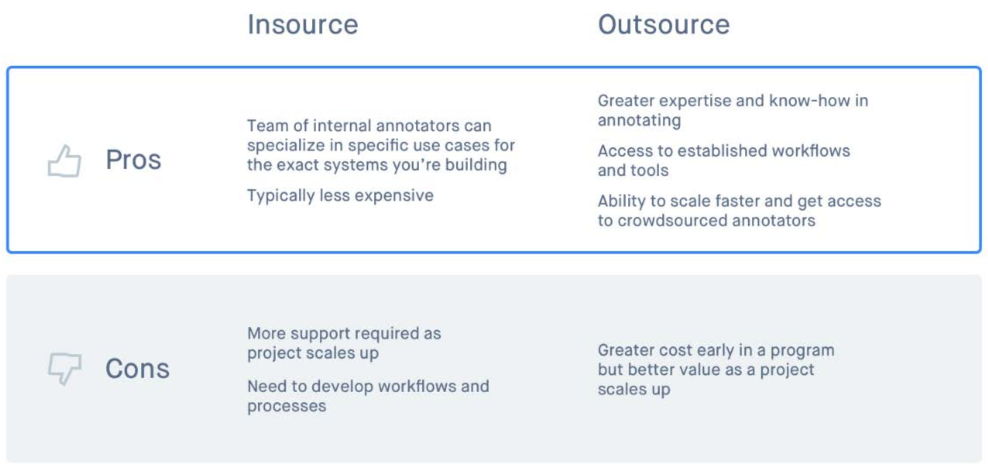
To avoid these issues, the data must first be curated. While massive amounts of data can be extremely valuable, mountains of the wrong data are worse than useless. In developing automated driving systems, it is important to capture as many anomalous scenarios and edge cases as possible. Data from a vehicle driving hundreds of miles down a straight highway across the middle of the US may offer far more limited information than just a few miles in a dense urban or even suburban area which are likely to provide more data about interactions with other road users. Conversely, rural driving may have fewer (but different) interactions, while providing views of aging infrastructure that need to be understood and weather patterns such as how snow blowing across a road can make it virtually invisible. Each has value, but it may be necessary to prioritize the data collection depending on how the vehicle is intended to be used. A geofenced urban robotaxi does not need as much emphasis on exurban driving.

Once relevant data has been identified, all crucial features in that data must be labeled and annotated. This is critical to making such datasets useful in developing a product. Labeling and annotating data may involve drawing bounding boxes around pedestrians, segmenting roads, creating polylines to denote road lines, using key points to mark where tires touch the ground, or a whole host of other annotation types to identify and track relevant objects in the data. Getting the most value from the data requires an understanding of the identity and the context. To train a prediction engine, it is crucial to be able to track specific objects from frame to frame to help understand where they are going and how they are likely to respond to other targets. For example, seeing how a pedestrian or cyclist responds over time to other vehicles or traffic signals can help a deep learning system understand the typical behavior in an area.

In-Source versus Outsource

As in so many areas of any business, build or buy is a crucial question. Even the largest organizations have limits on their resources. Given the importance of high quality data in so many modern applications, the question of whether to in-source or outsource is crucial. Handling data labeling and annotation requires expertise and processes.

For smaller companies and startups, there often is not much of a choice to be made. If the resources are not there to do the job, focusing on the core business is the direction that needs to be taken. Everything else goes to third-party vendors. Even for a large organization like an automaker or insurance company, the choice is not always obvious. While an automaker may have the financial resources to spin up a data operation, it takes time to find the people with the right expertise and set up processes. When pursuing a new technology area such as automated driving, even a large organization may be better off outsourcing this crucial component of the effort.



	Insource	Outsource
Pros	Team of internal annotators can specialize in specific use cases for the exact systems you're building Typically less expensive	Greater expertise and know-how in annotating Access to established workflows and tools Ability to scale faster and get access to crowdsourced annotators
Cons	More support required as project scales up Need to develop workflows and processes	Greater cost early in a program but better value as a project scales up

Source: Mighty AI

A company that is more familiar with this sort of data handling requirement, like an insurance company or large mapping provider, may be more capable of handling the job internally. However, it may be worthwhile for companies to consider having a specialist vendor take on the responsibility as data annotation and processing needs grow with new applications. A vendor that is focused on this area can bring new techniques and knowledge to the problem based on the broader scope of work being done, while also potentially providing a more efficient approach.

Human versus Machine

By nature, data labeling efforts invariably begin as a labor intensive, manual effort. The people doing the initial tagging need to go through a tedious process of examining the data to first curate the segments that are most relevant to the application. This process can be sped up significantly by having those that collect the data also capture time stamps as testing happens and as important anomalies are observed. However, even this is often insufficient to get every important bit of information. It also does not help when the data gathering is happening through an automated simulation process which may still require manual analysis.

For certain types of applications, once an ML system has begun to mature, it can be used to execute the first pass through the data. This can dramatically speed up the process of identifying targets and labeling them. An example of this is for mapping applications where machine vision can be used to identify lane markings, traffic signals, bridges, and other road features. As with human labeling, this would typically occur asynchronously after the data collection has been completed and it still requires human quality assurance to verify that the labeling is correct and nothing has been missed.

Even data intended for use in automated driving development or other applications can be run through automated labeling processes first once the models have sufficiently matured. However, the curation process of selecting data to be used for testing is still best left to human evaluation.

Most companies involved in automated vehicle development assign two people to each vehicle when running tests on the road. The vehicle operator typically has the sole task of supervising the vehicle, with hands near the steering wheel and feet near the pedals, ready to take over whenever it appears that the vehicle may not handle a situation correctly. These manual disengagements are logged, and companies doing on-road testing in California are required to file an annual report of all such incidents. Despite the inconsistent nature of how individual operators may decide to intervene, this is currently the only mandatory metric for evaluating automated vehicle reliability.

The automated flagging of any incident where the human operator takes control is a straightforward means of capturing scenarios that need further investigation. Even where there is no disengagement, automated flagging of situations for further analysis can be done by using criteria such as proximity to surrounding objects and closing speed, to detect near misses or even full contact.

In situations where there is no immediate safety risk, an operator may opt not to disengage but allow the system to continue so that further data can be collected about the system behavior. In this scenario, the second vehicle occupant, typically either a trained technician or engineer, can take note of the time and place where behavior occurred for later investigation. This manual flagging can be used at any time during testing.

Quality Is Job One

Even more important than how much data is available for training, simulation, and validation purposes is the data's quality. For example, if a dataset were provided to train a perception system that had trees labeled as pedestrians while pedestrians were labeled lamp posts, the algorithm would mis-recognize everything. Without an accurate understanding of its environment, an automated driving system could make the wrong decisions about what to do and potentially cause accidents rather than avoid them.

At this stage, with no real regulations in place anywhere in the world for automated vehicles and no significant commercial deployments, there is not yet a clear definition how good is good enough for DNNs. Thresholds will likely depend on the applications and the consequences of error. For example, a facial recognition error in something like a photo library application would simply be an annoyance, so the threshold for acceptability may be 50% accuracy or less. On the other hand, comparatively easy automated vehicle recognition problems such as the color of lane markings and whether they are solid or dashed could lead to tragic results if a vehicle decides to cross a line it should not. In this case, accuracy greater than 99% is almost certainly needed.

Even a more mundane application such as an insurance risk model could result in drivers being charged too much or too little if the data used to train the model is poorly annotated. Those doing the annotation need an understanding of the application the data will be used for and what elements of that data are most important. Robust annotator training, attention to detail, and quality assurance processes are essential.

Evaluating Data Quality

Data diversity is necessary not just for training of ML systems but also for validation of both the models and the training data itself. Initial evaluations of the data quality will generally be a manual process of human inspection. This technique can provide an indication of the completeness and relevance of a given dataset to ensure that it will provide the necessary coverage of scenarios.

However, given the sheer quantity of data used in contemporary systems, human inspection has scalability limits when it comes to evaluating the details of a dataset. As datasets grow, the percentage of data that gets human inspection may drop to statistical audit levels in the low single digits. However, there are mechanisms to automatically validate data robustness.

One method is to use a known gold standard DNN model that has been previously verified to give accurate results. Feeding a new dataset that has been labeled and annotated into such a model should yield an expected set of results. Any deviations from the expected results would be flagged as anomalies to be followed up on. Repeated runs can provide indications of consistency in the full system.

Ultimately, as development of any system matures, it is important to recheck old data to examine the reproducibility of results. Behavioral changes in the system are another red flag to developers to go back and take a deeper look to understand if the problem is in the data, the system, or both.

This is all particularly true when using synthetic data. As the data is modified from its source, it is crucial to be aware of any digital artifacts that may have been introduced into the system since these can throw off a ML algorithm. Automated processes for testing are necessary to reach scale, such as testing millions of miles per day on automated vehicles, but regular human audits of the results are essential to keeping the system honest.

HOW DO YOU PROVE THE ALGORITHM?

In classical software design, validation was a relatively straightforward process. The code generally had a structured set of instructions, and flow paths could be followed either by human reviewers or—in the case of more complex applications—by automated tools. As developers have learned painfully over the years, even that style of software could never be guaranteed to be free of errors, especially as complexity increased. Typically, the problem set being solved by software was constrained enough that a thorough testing suite in a combination of real environments and simulation could be used to validate virtually all of the functionality of the system.

Modern ML approaches to software are far more challenging to validate. The very nature of a trained neural network makes it far more difficult to evaluate directly in the way static code checks traditionally have been done. The potential operating environment for an automated vehicle is virtually limitless when accounting for road and infrastructure conditions, weather, time of day, location, and other road users.

As of mid-2018, there are no recognized standards for evaluating the efficacy of automated driving systems either on the road or in simulation. Every developer has their own evaluation criteria. The only commonly used metric is the rate of manual disengagements which companies testing in California must report. However, there is no industry standard for when a vehicle operator must take control from the automation, and the results are not comparable across companies.

At the 2018 Center for Automotive Research Management Briefing Seminars, Frank Mancheco, the Chief Product Officer at SAE International, announced that the engineering organization would begin a process of defining standard test procedures for automated vehicle evaluation. An initial framework is expected to be published by the end of 2018. SAE does not plan to develop acceptance criteria; that will be left to the industry and regulators. However, at least having a common measuring processes will allow for comparisons.

Until those testing procedures have been published, every company must develop its own suite of tests that cover a comprehensive set of use cases where the vehicle is expected to be deployed. This will range from unit and subsystem tests of the sensing and perception systems to full system evaluation. As an example of how this may be achieved, Argo AI has architected its automated vehicle software using a modular decomposition approach. Functions are broken down into the smallest practical chunks for unit testing. In this way—even where DNNs are used—if a module does not function properly, the issue can be isolated to a manageable part of the system rather than dealing with a single massive black box.

Common Standards

With any technology development, standardization is often both a long-term goal and one of the biggest challenges. Early in the life of a technology—when many parties are attempting to understand and solve the problems—standards are not even possible in many cases. At this stage it is often difficult to even articulate the precise requirements of all elements in the system.

With different groups having different aims, premature standardization efforts often lead to results that are overly broad and ultimately not useful. Such is the case with the SAE J3016 standard "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles." Each of the levels are vague enough that there is confusion about where any particular system fits within the levels.

In terms of data, its uses are so varied that any attempt to develop standards are unlikely to lead to any useful result.

Data Formats and Normalization

Among the more vexing challenges of commonizing and standardizing data are the formats and normalizing. Standard data formats are a more easily resolved issue and include factors such as how labels, annotations, and bounding boxes are represented in the files. As long as the format details for any given dataset are published, a translator can be developed to rewrite the file as needed. Elements such as data rates and resolution also need to be resolved to match what is expected from hardware sensors for use in simulation environments.

When using data in the development of applications such as automated driving, normalizing the data is also critical. Depending on how the sensors are configured on the vehicle, the view and perspective changes. The system using the data needs to understand where the view for any given sensor is coming from in order to understand where the detected objects are in real space. Similarly, if data from multiple sensors is being fused, the system needs to understand the relative perspectives of those sensors to align everything. Generic data may not contain this information and varied perspectives may show different relative positions of objects.

Community Data Pools

Given the need for diverse sources of data for both training and validation of ML systems, the concept of community data pools holds significant appeal for many developers. Such data collections have long been available from a variety of sources for academic and research purposes. The downside of community data is that it tends to be both more generic and more limited in scope than many developers would need for developing a production-ready system.

One challenge for using public datasets is the associated licensing. Public datasets may have specific restrictions on how they can be used, such as for research only or being completely in the public domain. Since there is a significant cost associated with producing a dataset, some contributors may require a fee for any uses that includes commercial applications. Other licenses may require that any new data generated through the use of a dataset is ultimately contributed back to the community. Some of the licenses currently being used for image datasets including the Gnu Public License, Creative Commons, and Open Data Commons Open Database License. Anyone planning to use or contribute to a community data pool should examine the licensing before beginning.

The biggest issue is that most of the companies working on production-intent applications are reluctant to share detailed data in case they reveal some trade secrets about how they are actually implementing their products. In the automated driving space in particular, companies are spending billions of dollars on the technology development and almost everyone is reluctant to surrender any potential competitive advantage. Conversely, using data without understanding its full origin may mean spending additional resources evaluating it and perhaps making corrections before it can be applied to a specific application.

Getting Contributions and Using Them

The dominant use of contributed data today is for research and academic purposes where there may be insufficient resources to conduct a full-blown data collection effort. Collecting enough data to be useful in training a robust ML system can take a significant amount of time and costly equipment that may not be available. Even for large companies involved in the development of automated driving it can take weeks or months with a fleet of vehicles to collect enough data to begin training systems and building high definition maps to enable testing.

For example, when a company like Argo AI expanded its development program with Ford into Miami in spring 2018, weeks were spent collecting data for the HD maps. This required driving all roads in the operating area multiple times from each lane to compile data about how each lane is used, capture data about occlusions such as buildings, overpasses, and signs from multiple perspectives and many other details. This data was then labeled, processed, and converted into detailed machine-readable maps that are used for localization and path planning.

For researchers or startups, there are both open-source and commercial datasets available that can be licensed and put to use. However, these datasets may also require significant modification or adaptation before they can be used for a specific application.

Crowdsourcing of data is another option to companies getting started. Silicon Valley startup Comma.ai is one example of a company that relied on a community to help gather data to be used for training its ML system. The company published an app that users could install on a smartphone mounted to allow the camera to look through the windshield as they drive. Video footage captured by the app would be saved and transmitted back to Comma servers where it could be utilized for development.

While this approach can generate massive quantities of raw data, it is not curated and still needs to be analyzed and labeled. By relying on a community, there is no guarantee that sufficient useful data will be collected. A refinement of this approach is for the developer to define the areas they need data from, geofencing the collection process. For example, a developer interested in getting data from San Francisco could have the crowdsourcing app only capture and transmit data from that region. Similarly, an app could provide users with suggested driving routes to collect even more granular data from areas that are likely to be more challenging for a system to work in.

Another downside to crowdsourcing relates to the problem of normalizing data. Since there is no control over what type of device is used, what its condition is or how it is mounted in the vehicle, the usefulness of the data may be limited. It may be usable for training the object recognition engine but not for path planning and control.

CONCLUSIONS

Ultimately data is at the heart of much of modern electronic system development, especially automated driving systems. Whether it is used for building, training, or validation, comprehensive and high quality datasets are essential. The output is only as good as the data fed into the system and low quality data can easily turn an otherwise viable product into something that may be worse than useless.

The nature of DNN systems will require billions of equivalent miles of simulation to validate safety and functionality, especially when used to power automated driving. That will require a mix of both real-world and synthetic data to provide adequate coverage and ensure system robustness.

Having dedicated teams of specialists that can collect, analyze, curate, annotate, and label data is key to successfully using the data to produce an excellent product. A team focused on the production of such datasets can use its expertise and established workflows to provide more efficient processing for the end users. Working with requirements from the end users, dedicated teams can also build the synthetic data required for development and validation. Any or all of these steps may be conducted internally of outsourced as best suits the needs and capabilities of an organization. Wherever the work is done, the processes should incorporate thorough quality assurance practices. Many data-related projects have failed as a result of low quality inputs. Considering the consequences, failure is not an option when it comes to automated driving.

Published 3Q 2018

Note: Editing of this report was closed on August 23, 2018.