brytlyt

The Rise of Accessible Data Science

A Whitepaper on Deep Learning for a Confident Future

Alex Harper

2022

Table of Contents

Summary

In 2021, the demand for data science skills hit a record high of 295% after a five-year climb. In 2020, IBM found that every person generated an average of 1.7 MB per second. For context, an audio call over your mobile phone uses around 2 MB per minute. With volumes of data booming like this and a persistent skill shortage, a large gap exists that requires specialist solutions. Traditional machine learning methods are not able to keep up. Deep learning is a more powerful solution that can handle larger datasets and does not require a trade off in performance as the scale of data increases. Deep neural networks can ingest and process unstructured data, speed up the pre-processing of structured data and automate feature extraction. This also reduces the likelihood of needing human intervention. However, the adoption of deep learning methods are best implemented when they are accompanied by a solution that mitigates the need for deep tech expertise.

80% of the cost of deep learning is incurred in deploying and keeping a model in production.

Until now, deep learning solutions have been fragmented with limited potential. In addition to the specialist knowledge required for building neural networks, deep learning is often deployed with multiple software packages across disparate hardware and different operating systems. Not only is this slow, but deploying a trained deep learning model eats up roughly 80% of the cost of deep learning. This happens because of the complex process behind keeping models up to date and relevant.

The Brytlyt platform addresses issues across the data science life cycle, including deep learning. From data discovery, data preparation and pre-training to training, deploying, through to maintaining MLOps (Machine Learning Operations) efficiency. With an advanced in-database AI solution built on PostgreSQL and PyTorch technology, Brytlyt can seamlessly slot into pre-existing tech stacks whilst supporting deep learning, traditional machine learning and external frameworks.

This paper will outline some of the challenges organisations face in adopting deep learning as they face a future driven by the advances in AI. It will then explore solutions available to users of the Brytlyt platform.

About Brytlyt

Brytlyt has built the world's first all-in-one serverless platform that brings in-database AI to GPU-accelerated analytics and visualisation. Instant insight and effortless deep learning are at the core of the platform. It should not take hours to query a complex dataset, and it should not take the best part of a week to train an intuitive AI model – Brytlyt's purpose is to challenge these notions. The platform also aims to make maintaining your models after deployment as simple as possible.

In-database AI

In-database AI aims to create a better experience for data scientists when building machine learning and deep learning models. The Brytlyt database uses integrated AI to ingest, explore, analyse, and visualise complex datasets in milliseconds, ready to launch a model into production.

Users can use in-database AI to train a model directly from their database without leaving the GPU-accelerated environment, removing the need for slow back and forth transfers of data between the database and the training environment.

Users can use in-database AI to train a model directly from their database without leaving the GPU-accelerated environment, removing the need for slow back and forth transfers of data between the database and the training environment.

Bringing AI to the Database

Any model that makes it into production must be wrapped in a solution that can do these three things:

- Authentication, user access, and security
- Scaling and concurrent usage
- Recording and tracking the production pipeline end-to-end and retaining the integrity of model versioning

Up until now, the approach has been to reinvent the wheel and build a solution from scratch that uses REST API, Python and microservices. But databases have already solved this challenge. By bringing AI to the database, your end solution is simple, cost effective, and reliable.

Why Deep Learning?

Deep learning is a sub-division of machine learning. It aims to mimic the neural pathways of the human brain with multiplex deep neural networks. The structure of an artificial neural network (as shown in Figure 1) has multiple hidden layers that inform the output. The properties created by the interactions between these hidden layers make deep learning a self-learning algorithm.

Traditional machine learning is a sub-division of AI (Artificial Intelligence). Algorithms are used to perform a task automatically – like predicting an outcome – by recognising patterns and relations in a dataset. Machine learning relies on a traditional statistical approach and requires more human input in the training process.

Deep learning produces highly intelligent results because it mimics the behaviour of human neural pathways. However, a major factor preventing organisations from implementing deep learning is a lack of the right skillset required to support it.

This has led to a rise in popularity of serverless deep learning platforms like Brytlyt that can make deep learning accessible to a wider variety of users.

Deep learning algorithms can not only do anything machine learning can do, they can also perform well on larger datasets. Where traditional machine learning requires a high level of human input from a skilled data engineer to extract features, a deep learning algorithm can do this on its own. This is a fundamental asset of the training process and is what makes deep learning so powerful

Where deep learning can handle billions of rows of data at a time, traditional machine learning algorithms are quickly overwhelmed as the amount of data grows. A major triumph of deep learning is extracting and learning features from raw, unstructured data with limited human instruction.

The challenges

There are two main issues affecting the widespread adoption of deep learning.

1. The level of skill and expertise needed to develop an effective neural network

2. The quantity of data needed to learn accurate patterns

Deep neural networks have a complex internal structure that needs to be fed more data than your standard machine learning algorithm for it to be self-sustaining.

Conversely, traditional machine learning requires input from a data engineer.

This is not to say that deep neural networks can or should be left ungoverned. Data is always evolving. Metrics must be defined, and model outputs must be tracked consistently to ensure accuracy and account for context. It is worth keeping in mind that whilst the data science journey ends with a fully trained model, launching that into production and maintaining it is a significant undertaking in its own right.

Does deep learning always require large datasets?

There is no straightforward answer. Typically, a neural network is more complex than a traditional machine learning model, but they can often be very similar.

A single neuron is all that is needed to implement a neural network for linear or logistic regression. But in technical terms, it would not be a true neural network with less than three hidden layers. The complex nature of a deep learning model comes with a corresponding need for more data, the greater the complexity, the more data is needed.

The amount of data required for successfully training a model using deep learning depends on at least two factors

1. The complexity of the problem statement.

2. The complexity of the deep learning algorithm.

Because of this, it makes sense to start with a simple use case that can be iteratively developed into a more complex solution as needed.

The Nature of Data

Data is available to us 24/7, and this is proving a challenge for traditional technology solutions. Applications like Brytlyt that use real-time streaming of data are becoming more prevalent, so any solution that implements deep learning or MLOps needs to be able to handle data-in-motion.

Structured and Unstructured Data

Structured data is arranged in tables with rows and columns of numerical values, dates and strings. The best example of where we see structured data is in a relational (SQL) database like the one shown in Figure 2.

The uniformity of structured data means statistical analysis is easier to perform when compared to unstructured data. It is used in machine learning algorithms, including linear and logistic regression, random forest, naïve Bayes, clustering analysis, and gradient boosting too.

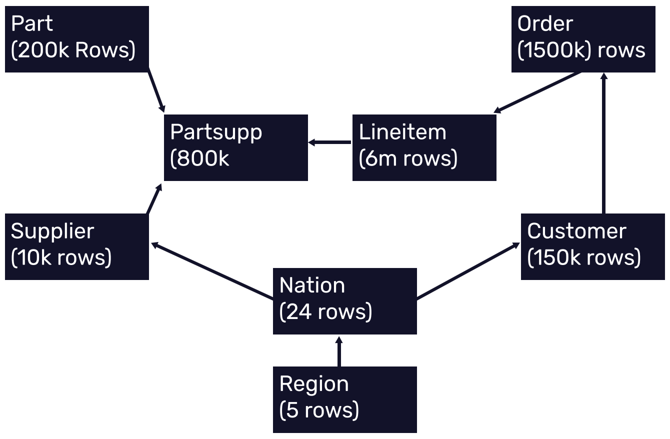

Figure 2: An example of a Relational Database

The volume of unstructured data exceeds the volume of structured data, but is more subjective in its use. Unstructured data can be made up of anything from audio to emails and blogs. It makes up around 80% of the volume of data used.

Feature Engineering and Data Discovery

Feature engineering was once a complex manual task that required experienced data engineers to manipulate and transform raw data. Now, it can mostly be automated, and critical features can be extracted from raw data and manipulated in real-time.

50% of a Data Scientist's time is spent in exploratory data analysis alongside feature creation and transformation.

One goal of feature engineering is to create new variables that are not in the initial training set but are still related to it, also known as feature transformation. The aim of this is to simplify and speed up data transformations whilst ensuring that the model is as accurate as possible. Regardless of the data or model architecture, a badly engineered feature will always have a negative impact.

Most of this work is done in a database, which is why the majority of time spent by data scientists is using SQL and database. No matter how well formatted a data set is, it is likely that it will need work to prepare it for machine learning algorithms. There are three ways to do this.

Exploratory data analysis

An important first step in any machine learning process is exploratory data analysis, so that we can better understand its most basic properties. Often this involves using simple algebra rules or standard machine learning models to piece together the story that the data is telling. Some of these techniques include: normalisation and scaling or linear and polynomial regression.

Feature creation

Creating features involves creating new variables that are the best fit for any given model. As a basic example, the value of a house may be partially determined by the length and width of the land it is built on. A feature we could create for this then, would be:

$$\text{area} = \text{length} \times \text{width}$$

Feature transformation

A feature transformation is any function – normalisation, scaling, or adding constants – that transforms a feature from one classification to another. The dataset is altered but it retains its information properties. A common example of feature transformation is scaling. By transforming features, the number of features used in the training set can be scaled up or down, thus providing an increase in the speed of training and/or the accuracy of a given model.

For example, if there is a one-line address column (one string), the feature can be transformed to create additional relevant features. In other words, it can be scaled up.

Address
P. Sherman, 42 Wallaby Way, Sydney

Becomes:

Name	Address	City
P. Sherman	42 Wallaby Way	Sydney

Figure 4: An example of feature transformation

Low-Code Deep Learning

Low-code machine learning is an emerging trend in the AI space. For the practice of deep learning to be made more accessible to a larger population of users, the tools used to build deep learning algorithms need to be simplified and easy to use.

By using a platform like Brytlyt that contains a library of predefined models, it is possible to access the power of deep learning through a design 'n' build framework.

What is a predefined model?

A predefined model is a pre-built but untrained neural network. Brytlyt has its own library of pre-built, untrained and some partially trained neural networks. Predefined models can form the basis of an Integrated Development Environment (IDE), and enable a low-code environment for more general use.

Integrated Development Environments

Platforms like Brytlyt provide an IDE for creating and deploying deep learning models. Built by experts in the field and equipped with the right tools, users can build complex models faster, deploy them sooner and maintain them with less overhead.

For example, a study may require you to classify mushrooms into edible or poisonous by species. Using a database of mushrooms features that includes size, weight, stalk length, cap diameter, cap colour, and cap shape, a multi modal approach can be applied.

- A binary classification problem to sort data into edible and poisonous mushrooms.
- Two multi classification problems to sort the data into its respective species.

In this approach, the first building block we need is the binary classifier (edible or poisonous) followed by two additional separate building blocks to predict its species. The final building block would be the result of edible or poisonous, plus its species variant.

In this example, the binary classifier, two multi-classifiers, and the concatenation are each four individual predefined models that can be 'dragged and dropped' into the desired order.

Serverless Platforms

Cloud computing provides incredible flexibility to access compute and storage infrastructure. But the legacy software running on this infrastructure was not designed to support the on-off profile that a serverless pay-as-you-go needs, so typically requires hardware to be up and running constantly.

Serverless computing combines hardware, software, user session state and data using an on-demand 'pay-as-you-go' method. This all happens when the user connects to the application.

When the user is disconnected, the hardware is made inactive by triggering an 'on-off' switch within the software. The framework of serverless computing is cloud-native so can be used to build and run applications without having to manage or maintain hardware, install software or manage security. A serverless solution requires a rewrite of the application software so that it can rapidly switch between an active and inactive state.

Besides up to 90% lower cost and an individually economical approach to the physical resources required, other benefits of serverless platforms are:

- Load balancing and capacity management is automated
- Software updates and security patches are immediately available
- Effortless scaling up or down depending on usage

Using serverless* computing means resources can be allocated and scaled on demand, and only used as needed.

*On-premise options are available.

Brytlyt's Integration with Open-Source Software

The Brytlyt platform is tightly integrated with a huge array of open-source software like PostgreSQL, PyTorch, Jupyter Notebooks, Scikit-learn, and MLflow amongst others. The ability to manipulate code fit for a unique purpose is invaluable to data scientists and data engineers. They are also completely free and are often at the forefront of modern technology.

PostgreSQL

The Brytlyt database is based on PostgreSQL technology. In addition to the framework being familiar, it is already compatible with a range of existing systems and well-known platforms.

Brytlyt has the added benefit of an easy-to-use enhanced version of the standard PostgreSQL connector used by Python. The connector allows the user to perform SQL queries and transfer data using python syntax, which is then executed on GPUs.

PyTorch

Brytlyt has a unique integration with PyTorch that works to translate data as tensors and store data as columns in tables in the PostgreSQL database, without leaving the GPU environment. This capability effectively bridges the gap between the world of the database and the data science world; in this instance SQL and PyTorch.

Scikit-learn

Scikit-learn-learn is a python library for machine learning. It has fast and efficient tools for machine learning and statistical modelling, such as classification, regression, and clustering. It also has a multi-layered perceptron model for deep learning. Brytlyt's integration with Scikit accelerates the process.

NumPy and Pandas

NumPy and Pandas are both libraries designed for seamless data processing. Brytlyt users are not held up with tedious data processing tasks.

Jupyter Notebooks

Brytlyt works with Jupyter to support 100+ coding languages.

MLflow

MLflow is a tool designed to manage the machine learning lifecycle from end-to-end, making MLOps a seamless process.

The Purpose of MLOps

MLOps is the process aimed at unifying the iterative maintenance process following model deployment. Monitoring and maintenance can be done using tools designed for MLOps (like MLflow). Solutions built around the MLOps framework aim to function automatically with limited intervention.

AI models lose their integrity rapidly due to data drift and a constantly changing environment. The MLOps process should improve the quality of production models whilst retaining validation, model and data versioning, testing and regularisation.

The following are the stages included in an effective MLOps system.

Deployment

The final stage of the data science pipeline and first stage of MLOps is deploying models to the production system. This is traditionally handled manually by selecting the best model from the array of experiments and then pushing it into production. For example, a target metric like accuracy, precision, or recon recall, can be selected so that the model that performs the best with respect to that target can then be pushed to run in production. When implementing MLOps, this is done automatically.

Framing the problem (CI/CD)

Because any model that reaches production immediately begins to age, continuous integration allows for new features and improved models to reach production faster and more reliably. As data ages it becomes less able to reflect a real-world environment. As a result, the model's performance can suffer, dwindle, or even become irrelevant.

There are two options when this happens.

1. New data can be added to the historical data to fortify the model in retraining across the whole data set, or...

2. The model can be trained on the new data only to refresh the model to the current real-world environment

In any case, monitoring and retraining is a basic requirement to keep these models relevant. Changes in the summary statistics of the data, including calculations of the data count, mean, median, and standard deviation, can give vital clues as to the reasons why the effectiveness of an algorithm can decay. This needs to be monitored side-by-side with the model's effectiveness so that the model can be updated or changed accordingly.

The Future of Deep Learning is Bright

There is no doubt that deep learning has the potential to transform the way we use data. However, we have established that, traditionally, deep learning has been a laborious process. The investment required to design, build and then deploy an artificial neural network for a model to train against, is an intimidating concept for many organisations in major industries – such as Oil and Gas, Financial Crime and Telecoms – that want to evolve AI pipelines.

AI and actionable intelligence rely on the accessibility and speed of data science enabled platforms. These platforms now incorporate cutting edge tools designed to simplify the way data scientists and business intelligence professionals work.

Brytlyt brings data science, analytics, and visualisation together in one platform to tackle these challenges. Organisations around the globe using Brytlyt will see the benefits of an integrated data science solution powered by GPUs, alongside powerful in-database AI features such as instant Tensor-to-SQL table translation for accessibility, bridging the gap between SQP and PyTorch