Deep Learning and the Artificial Intelligence Revolution

August 2017

Table of Contents

Introduction 1

History of Artificial Intelligence 1

Why Is AI Taking Off Now? 2

Differences Between Artificial Intelligence, Machine Learning, and Deep Learning 3
- Supervised vs. Unsupervised
- What is Deep Learning?
- How Does Training Work?

Database Considerations with Deep Learning 6

Why MongoDB for Deep Learning? 7
- Flexible Data Model
- Rich Programming and Query Model
- Performance, Scalability & Redundancy
- Tunable Consistency

MongoDB AI Deployments 9

Summary 10

We Can Help 11

Sources 11

Introduction

We are living in an era where artificial intelligence (AI) has started to scratch the surface of its true potential. Not only does AI create the possibility of disrupting industries and transforming the workplace, but it can also address some of society's biggest challenges. Autonomous vehicles may save tens of thousands of lives, and increase mobility for the elderly and the disabled. Precision medicine may unlock tailored individual treatment that extends life. Smart buildings may help reduce carbon emissions and save energy. These are just a few of the potential benefits that AI promises, and is starting to deliver upon.

By 2018, Gartner estimates that machines will author 20% of all business content, and an expected 6 billion IoT-connected devices will be generating a deluge of data. AI will be essential to make sense of it all. No longer is AI confined to science fiction movies; artificial intelligence and machine learning are finding real world applicability and adoption.

In this white paper we will explore the history of AI, why AI (and particularly deep learning) is becoming pervasive today, and how deep learning works. We will also discuss the considerations developers and data scientists need to be aware of when selecting the appropriate databases for storage of the underlying input, training, and results data. If you're already familiar with the evolution of AI and Deep Learning, head directly for the "Database Considerations with Deep Learning" section of the paper.

History of Artificial Intelligence

Artificial intelligence has been a dream for many ever since Alan Turing wrote his seminal 1950 paper "Computing Machinery and Intelligence". In Turing's paper, he asked the fundamental question, "Can Machines Think?" and contemplated the concept of whether computers could communicate like humans. The birth of the AI field really started in the summer of 1956, when a group of researchers came together at Dartmouth College to initiate a series of research projects aimed at programming computers to behave like humans. It was at Dartmouth where the term "artificial intelligence" was first coined, and concepts from the conference crystallized to form a legitimate interdisciplinary research area.

Over the next decade, progress in AI experienced boom and bust cycles as advances with new algorithms were constrained by the limitations of contemporary technologies. In 1968, the science fiction film 2001: A Space Odyssey helped AI leave an indelible impression in mainstream consciousness when a sentient computer – HAL 9000 – uttered the famous line, "I'm sorry Dave, I'm afraid I can't do that." In the late 1970s, Star Wars further cemented AI in mainstream culture when a duo of artificially intelligent robots (C-3PO and R2-D2) helped save the galaxy.

But it wasn't until the late 1990s that AI began to transition from science fiction lore into real world applicability. Beginning in 1997 with IBM's Deep Blue chess program beating then current world champion Garry Kasparov, the late 1990s ushered in a new era of AI in which progress started to accelerate. Researchers began to focus on sub-problems of AI and harness it to solve real world applications such as image recognition and speech. Instead of trying to structure logical rules determined by the knowledge of experts, researchers started to work on how algorithms could learn the logical rules themselves. This trend helped to shift research focus into Artificial Neural Networks (ANNs). First conceptualized in the 1940s, ANNs were invented to "loosely" mimic how the human brain learns. ANNs experienced a resurgence in popularity in 1986 when the concept of backpropagation gradient descent was improved. The backpropagation method reduced the huge number of permutations needed in an ANN, and thus was a more efficient way to reduce AI training time.

Even with advances in new algorithms, neural networks still suffered from limitations with technology that had plagued their adoption over the previous decades. It wasn't until the mid 2000s that another wave of progress in AI started to take form. In 2006, Geoffrey Hinton of the University of Toronto made a modification to ANNs, which he called deep learning (deep neural networks). Hinton added multiple layers to ANNs and mathematically optimized the results from each layer so that learning accumulated faster up the stack of layers. In 2012, Andrew Ng of Stanford University took deep learning a step further when he built a crude implementation of deep neural networks using Graphical Processing Units (GPUs). Since GPUs have a massively parallel architecture that consist of thousands of cores designed to handle multiple tasks simultaneously, Ng found that a cluster of GPUs could train a deep learning model much faster than general purpose CPUs. Rather than take weeks to generate a model with traditional CPUs, he was able to perform the same task in a day with GPUs.

Essentially, this convergence – advances in software algorithms combined with highly performant hardware – had been brewing for decades, and would usher in the rapid progress AI is currently experiencing.

Why Is AI Taking Off Now?

There are four main factors driving the adoption of AI today:

More Data. AI needs a huge amount of data to learn, and the digitization of society is providing the available raw material to fuel its advances. Big data from sources such as Internet of Things (IoT) sensors, social and mobile computing, science and academia, healthcare, and many more new applications generate data that can be used to train AI models. Not surprisingly, the companies investing most in AI – Amazon, Apple, Baidu, Google, Microsoft, Facebook – are the ones with the most data.

Cheaper Computation. In the past, even as AI algorithms improved, hardware remained a constraining factor. Recent advances in hardware and new computational models, particularly around GPUs, have accelerated the adoption of AI. GPUs gained popularity in the AI community for their ability to handle a high degree of parallel operations and perform matrix multiplications in an efficient manner – both are necessary for the iterative nature of deep learning algorithms. Subsequently, CPUs have also made advances for AI applications. Recently, Intel added new deep learning instructions to its Xeon and Xeon Phi processors to allow for better parallelization and more efficient matrix computation. This is coupled with improved tools and software frameworks from it's software development libraries. With the adoption of AI, hardware vendors now also have the chip demand to justify and amortize the large capital costs required to develop, design, and manufacture products exclusively tailored for AI. These advancements

result in better hardware designs, performance, and power usage profiles.

More Sophisticated Algorithms. Higher performance and less expensive compute also enable researchers to develop and train more advanced algorithms because they aren't limited by the hardware constraints of the past. As a result, deep learning is now solving specific problems (e.g., speech recognition, image classification, handwriting recognition, fraud detection) with astonishing accuracy, and more advanced algorithms continue to advance the state of the art in AI.

Broader Investment. Over the past decades, AI research and development was primarily limited to universities and research institutions. Lack of funding combined with the sheer difficulty of the problems associated with AI resulted in minimal progress. Today, AI investment is no longer confined to university laboratories, but is pervasive in many areas – government, venture capital-backed startups, internet giants, and large enterprises across every industry sector.

Differences Between Artificial Intelligence, Machine Learning, and Deep Learning

In many contexts, artificial intelligence, machine learning and deep learning are used interchangeably, but in reality, machine and deep learning is a subset of AI. We can think of AI as the branch of computer science focused on building machines capable of intelligent behaviour, while machine and deep learning is the practice of using algorithms to sift through data, learn from the data, and make predictions or take autonomous actions. Therefore, instead of programming specific constraints for an algorithm to follow, the algorithm is trained using large amounts of data to give it the ability to independently learn, reason, and perform a specific task.

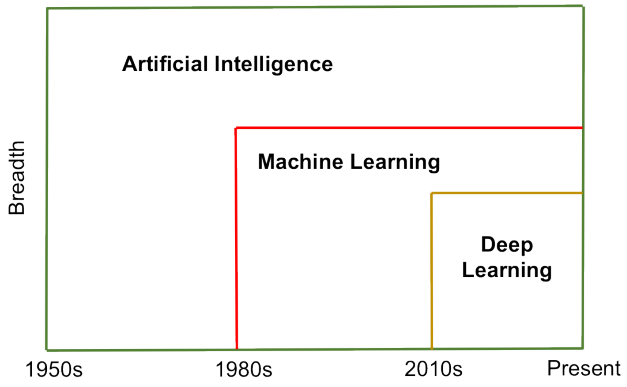

Figure 1: Timeline of Artificial Intelligence, Machine Learning, and Deep Learning

So what's the difference between machine learning and deep learning? Before defining deep learning, let's dig deeper into machine learning.

Supervised vs. Unsupervised

There are two main classes of machine learning approaches: supervised learning and unsupervised learning.

Supervised Learning. Currently, supervised learning is the most common type of machine learning algorithm. With supervised learning, the algorithm takes input data manually labelled by developers and analysts, using it to train the model and generate predictions. Supervised learning can be delineated into two groups: regression and classification problems.

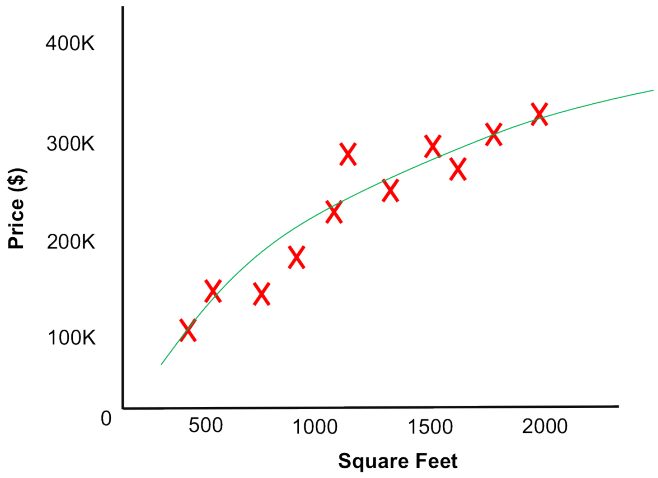

Figure 2: Supervised Regression Example

Figure 2 demonstrates a simple regression problem. Here, there are two inputs, or features (square feet and price), that are used to generate a curve fitting line and make subsequent predictions of property price.

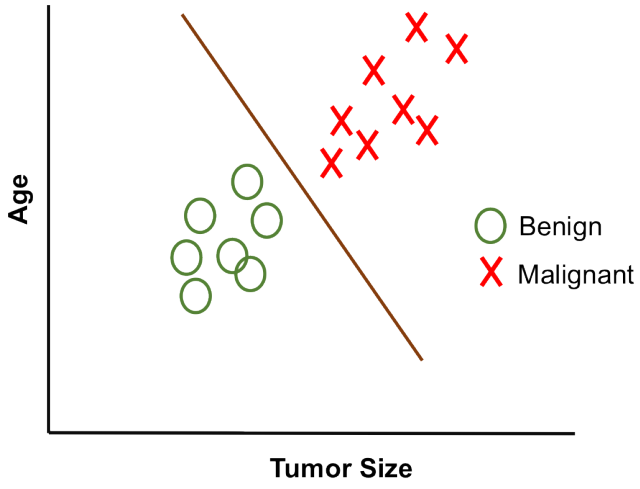

Figure 3: Supervised Classification Example

Figure 3 is an example of a supervised classification example. The dataset is labelled with benign and malignant tumors for breast cancer patients. The supervised classification algorithm will attempt to segment tumors into two different classifications by fitting a straight line through the data. Future data can then be classified as benign or malignant based on the straight-line classification. Classification problems result in discrete outputs, though that does not necessarily constrain the number of outputs to a fixed set. Figure 3 has only two discrete outputs, but there could be many more classifications (benign, Type 1 malignant, Type 2 malignant, etc.)

Unsupervised Learning. In our supervised learning example, labelled datasets (benign or malignant classifications) help the algorithm determine what the correct answer is. With unsupervised learning, we give the algorithm an unlabelled dataset, and depend on the algorithm to uncover structures and patterns in the data.

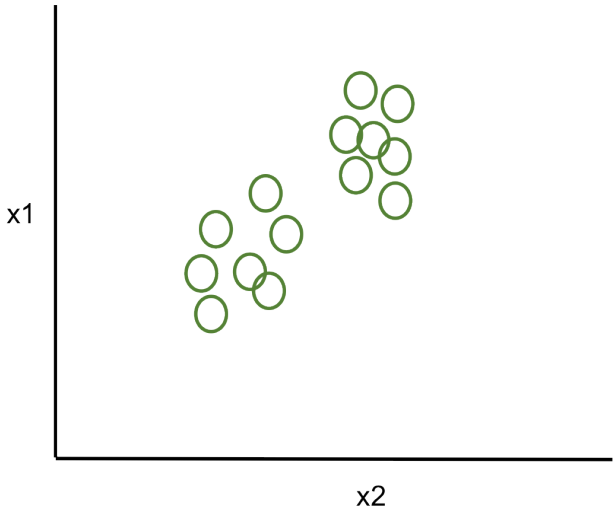

Figure 4: Unsupervised Learning Example

In Figure 4, there is no information about what each data point represents, and so the algorithm is asked to find structure in the data independently of any supervision. Here, the unsupervised learning algorithm might determine there are two distinct clusters and make a straight-line classification between the clusters. Unsupervised learning is broadly applied in many use cases such as Google News, social network analysis, market segmentation, and astronomical analysis around galaxy formations.

What is Deep Learning?

Deep learning is a subset of machine learning that has attracted worldwide attention for its recent success solving particularly hard and large-scale problems in areas such as speech recognition, natural language processing, and image classification. Deep learning is a refinement of ANNs, which, as discussed earlier, "loosely" emulate how the human brain learns and solves problems.

Before diving into how deep learning works, it's important to first understand how ANNs work. ANNs are made up of an interconnected group of neurons, similar to the network of neurons in the brain.

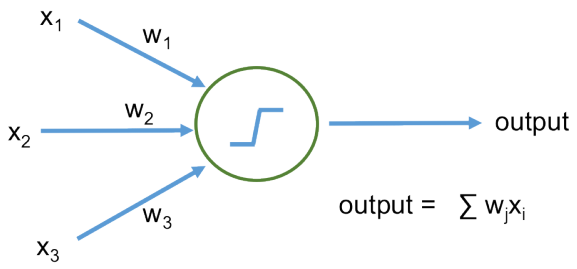

Figure 5: The Neuron Model

At a simplistic level, a neuron in a neural network is a unit that receives a number of inputs (xi), performs a computation on the inputs, and then sends the output to other nodes or neurons in the network. Weights (wj), or parameters, represent the strength of the input connection and can be either positive or negative. The inputs are multiplied by the associated weights (x1w1, x2w2,..) and the neuron adds the output from all inputs. The final step is that a neuron performs a computation, or activation function. The activation function (sigmoid function is popular) allows an ANN to model complex nonlinear patterns that simpler models may not represent correctly.

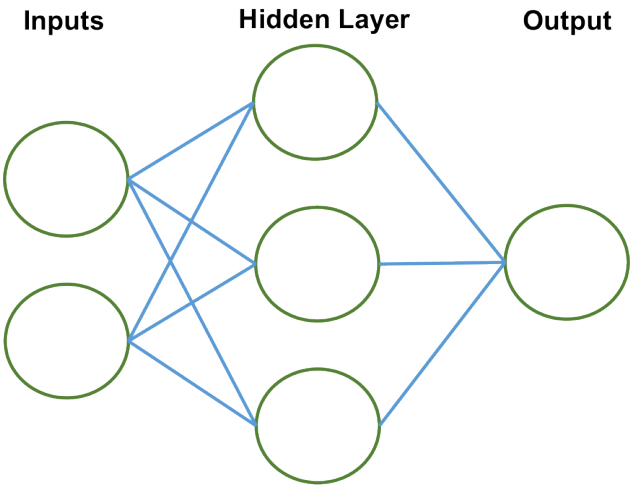

Figure 6: Neural Network Diagram

Figure 6 represents a neural network. The first layer is called the input layer and is where features (x1, x2, x3) are input. The second layer is called the hidden layer. Any layer that is not an input or output layer is a hidden layer. Deep learning was originally coined because of the multiple levels of hidden layers. Networks typically contain more than 3 hidden layers, and in some cases more than 1,200 hidden layers.

What is the benefit of multiple hidden layers? Certain patterns may need deeper investigation that can be surfaced with the additional hidden layers. Image classification is an area where deep learning can achieve high performance on very hard visual recognition tasks – even exceeding human performance in certain areas. Let's illustrate this point with an example of how additional hidden layers help perform facial recognition.

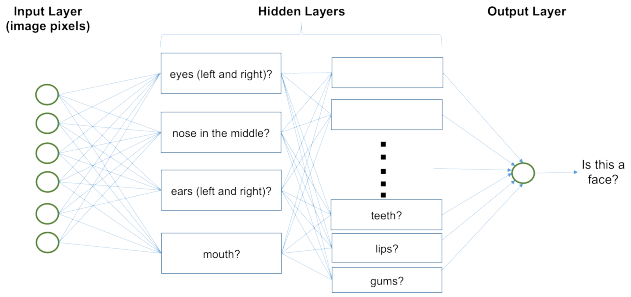

Figure 7: Deep Learning Image Recognition

When a picture is input into a deep learning network, it is first decomposed into image pixels. The algorithm will then look for patterns of shapes at certain locations in the image. The first hidden layer might try to uncover specific facial patterns: eyes, mouth, nose, ears. Adding an additional hidden layer deconstructs the facial patterns into more granular attributes. For example, a "mouth" could be further deconstructed into "teeth", "lips", "gums", etc. Adding additional hidden layers can devolve these patterns even further to recognize the subtlest nuances. The end result is that a deep learning network can break down a very complicated problem into a set of simple questions. The hidden layers are essentially a hierarchical grouping of different variables that provide a better defined relationship. Currently, most deep learning algorithms are supervised; thus, deep learning models are trained against a known truth.

How Does Training Work?

The purpose of training a deep learning model is to reduce the cost function, which is the discrepancy between the expected output and the real output. The connections between the nodes will have specific weights that need to be refined to minimize the discrepancy. By modifying the weights, we can minimize the cost function to its global minimum, which means we've reduced the error in our

model to its lowest value. The reason deep learning is so computationally intensive is that it requires finding the correct set of weights within millions or billions of connections. This is where constant iteration is required as new sequences of weights are tested repeatedly to find the point where the cost function is at its global minimum.

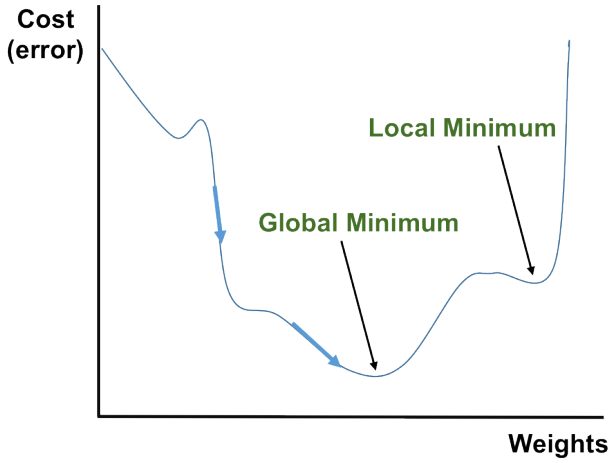

Figure 8: Deep Learning Cost Function

A common technique in deep learning is to use backpropagation gradient descent. Gradient descent emerged as an efficient mathematical optimization that works effectively with a large number of dimensions (or features) without having to perform brute force dimensionality analysis. Gradient descent works by computing a gradient (or slope) in the direction of the function global minimum based on the weights. During training, weights are first randomly assigned, and an error is calculated. Based on the error, gradient descent will then modify the weights, backpropagate the updated weights through the multiple layers, and retrain the model such that the cost function moves towards the global minimum. This continues iteratively until the cost function reaches the global minimum. There may be instances where gradient descent resolves itself at a local minimum instead of global minimum. Methods to mitigate this issue is to use a convex function or generate more randomness for the parameters.

Database Considerations with Deep Learning

Non-relational databases have played an integral role in the recent advancement of the technology enabling machine learning and deep learning. The ability to collect and store large volumes of structured and unstructured data has provided deep learning with the raw material needed to improve predictions. When building a deep learning application, there are certain considerations to keep in mind when selecting a database for management of underlying data.

Flexible Data Model. In deep learning there are three stages where data needs to be persisted – input data, training data, and results data. Deep learning is a dynamic process that typically involves significant experimentation. For example, it is not uncommon for frequent modifications to occur during the experimentation process – tuning hyperparameters, adding unstructured input data, modifying the results output – as new information and insights are uncovered. Therefore, it is important to choose a database that is built on a flexible data model, avoiding the need to perform costly schema migrations whenever data structures need to change.

Scale. One of the biggest challenges of deep learning is the time required to train a model. Deep learning models can take weeks to train – as algorithms such as gradient descent may require many iterations of matrix operations involving billions of parameters. In order to reduce training times, deep learning frameworks try to parallelize the training workload across fleets of distributed commodity servers.

There are two main ways to parallelize training: data parallelism and model parallelism.

- **Data parallelism.** Splitting the data across many nodes for processing and storage, enabled by distributed systems such as Apache Spark, MongoDB, and Apache Hadoop
- **Model parallelism.** Splitting the model and its associated layers across many nodes, enabled by software libraries and frameworks such as Tensorflow, Caffe, and Theano. Splitting provides parallelism, but

does incur a performance cost in coordinating outputs between different nodes

In addition to the model's training phase, another big challenge of deep learning is that the input datasets are continuously growing, which increases the number of parameters to train against. Not only does this mean that the input dataset may exceed available server memory, but it also means that the matrices involved in gradient descent can exceed the node's memory as well. Thus, scaling out, rather than scaling up, is important as this enables the workload and associated dataset to be distributed across multiple nodes, allowing computations to be performed in parallel.

Fault Tolerance. Many deep learning algorithms use checkpointing as a way to recover training data in the event of failure. However, frequent checkpointing requires significant system overhead. An alternative is to leverage the use of multiple data replicas hosted on separate nodes. These replicas provide redundancy and data availability without the need to consume resources on the primary node of the system.

Consistency. For most deep learning algorithms it is recommended to use a strong data consistency model. With strong consistency each node in a distributed database cluster is operating on the latest copy of the data, though some algorithms, such as Stochastic Gradient Descent (SGD), can tolerate a certain degree of inconsistency. Strong consistency will provide the most accurate results, but in certain situations where faster training time is valued over accuracy, eventual consistency is acceptable. To optimize for accuracy and performance, the database should offer tunable consistency.

Why MongoDB for Deep Learning?

Developers and data scientists can harness MongoDB as a flexible, scalable, and performant distributed database to meet the rigors of AI application development. MongoDB is the only database that harnesses the innovations of NoSQL – data model flexibility and distributed, scale-out design – while maintaining the foundations of rich query capabilities and strong consistency that have made relational databases the platform for enterprise applications over the past three decades.

Flexible Data Model

MongoDB's document data model makes it easy for developers and data scientists to store and combine data of any structure within the database, without giving up sophisticated validation rules to govern data quality. The schema can be dynamically modified without application or database downtime that results from costly schema modifications or redesign incurred by relational database systems.

This data model flexibility is especially valuable to deep learning, which involves constant experimentation to uncover new insights and predictions:

- Input datasets can comprise rapidly changing structured and unstructured data ingested from clickstreams, log files, social media and IoT sensor streams, CSV, text, images, video, and more. Many of these datasets do not map well into the rigid row and column formats of relational databases.
- The training process often involves adding new hidden layers, feature labels, hyperparameters, and input data, requiring frequent modifications to the underlying data model.

A database supporting a wide variety of input datasets, with the ability to seamlessly modify parameters for model training, is therefore essential.

Rich Programming and Query Model

MongoDB offers both native drivers and certified connectors for developers and data scientists building deep learning models with data from MongoDB. The PyMongo driver is the recommended way to work with MongoDB from Python, implementing an idiomatic API that makes development natural for Python programmers. The community developed MongoDB Client for R is also available for R programmers.

The MongoDB query language and rich secondary indexes enable developers to build applications that can query and analyze the data in multiple ways. Data can be accessed by

single keys, ranges, text search, graph, and geospatial queries through to complex aggregations and MapReduce jobs, returning responses in milliseconds.

To parallelize data processing across a distributed database cluster, MongoDB provides the aggregation pipeline and MapReduce. The MongoDB aggregation pipeline is modeled on the concept of data processing pipelines. Documents enter a multi-stage pipeline that transforms the documents into an aggregated result using native operations executed within MongoDB. The most basic pipeline stages provide filters that operate like queries, and document transformations that modify the form of the output document. Other pipeline operations provide tools for grouping and sorting documents by specific fields as well as tools for aggregating the contents of arrays, including arrays of documents. In addition, pipeline stages can use operators for tasks such as calculating the average or standard deviations across collections of documents, and manipulating strings. MongoDB also provides native MapReduce operations within the database, using custom JavaScript functions to perform the map and reduce stages.

In addition to its native query framework, MongoDB also offers a high performance connector for Apache Spark. The connector exposes all of Spark's libraries, including Python, R, Scala and Java. MongoDB data is materialized as DataFrames and Datasets for analysis with machine learning, graph, streaming, and SQL APIs.

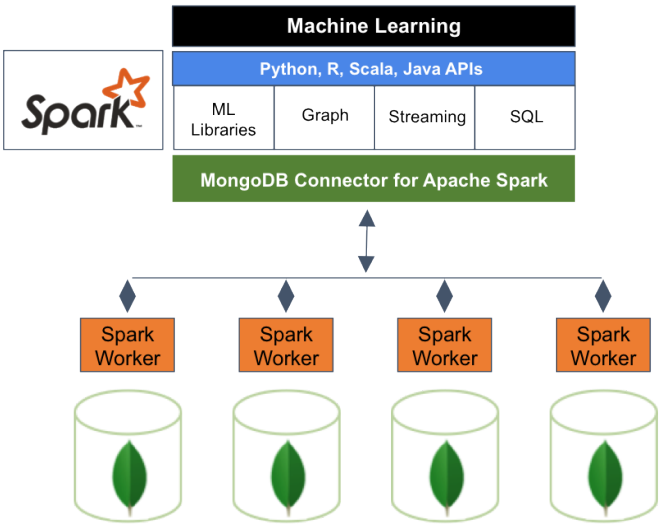

Figure 9: Combining MongoDB with Spark with Sophisticated Analytics & Machine Learning

The MongoDB Connector for Apache Spark can take advantage of MongoDB's aggregation pipeline and secondary indexes to extract, filter, and process only the range of data it needs – for example, analyzing all customers located in a specific geography. This is very different from simple NoSQL datastores that do not support either secondary indexes or in-database aggregations. In these cases, Spark would need to extract all data based on a simple primary key, even if only a subset of that data is required for the Spark process. This means more processing overhead, more hardware, and longer time-to-insight for data scientists and engineers. To maximize performance across large, distributed data sets, the MongoDB Connector for Apache Spark can co-locate Resilient Distributed Datasets (RDDs) with the source MongoDB node, thereby minimizing data movement across the cluster and reducing latency.

Performance, Scalability & Redundancy

Model training time can be reduced by building the deep learning platform on top of a performant and scalable database layer. MongoDB offers a number of innovations to maximize throughput and minimize latency of deep learning workloads:

- WiredTiger is the default storage engine for MongoDB, developed by the architects of Berkeley DB, the most widely deployed embedded data management software in the world. WiredTiger scales on modern, multi-core architectures. Using a variety of programming techniques such as hazard pointers, lock-free algorithms, fast latching and message passing, WiredTiger maximizes computational work per CPU core and clock cycle. To minimize on-disk overhead and I/O, WiredTiger uses compact file formats and storage compression.
- For the most latency-sensitive deep learning applications, MongoDB can be configured with the In-Memory storage engine. Based on WiredTiger, this storage engine gives users the benefits of in-memory computing, without trading away the rich query flexibility, real-time analytics, and scalable capacity offered by conventional disk-based databases.
- To parallelize model training and scale input datasets beyond a single node, MongoDB uses a technique

called sharding, which distributes processing and data across clusters of commodity hardware. MongoDB sharding is fully elastic, automatically rebalancing data across the cluster as the input dataset grows, or as nodes are added and removed.

- Within a MongoDB cluster, data from each shard is automatically distributed to multiple replicas hosted on separate nodes. MongoDB replica sets provide redundancy to recover training data in the event of a failure, reducing the overhead of checkpointing.

Tunable Consistency

MongoDB is strongly consistent by default, enabling deep learning applications to immediately read what has been written to the database, thus avoiding the developer complexity imposed by eventually consistent systems. Strong consistency will provide the most accurate results for machine learning algorithms; however in some scenarios, such as SGD, it is acceptable to trade consistency against specific performance goals by distributing queries across a cluster of MongoDB secondary replica set members.

MongoDB AI Deployments

MongoDB is serving as the database for many AI and deep learning platforms. A selection of users across different applications and industries follows:

IBM Watson: Analytics & Visualization

Watson Analytics is IBM's cloud-hosted service providing smart data discovery to guide data exploration, automate predictive analytics and visualize outputs. Watson Analytics is used across banking, insurance, retail, telecommunications, petroleum, and government applications. MongoDB is used alongside DB2 for managing data storage. MongoDB provides a metadata repository of all source data assets and analytics visualizations, stored in rich JSON document structures, with the scalability to support tens of thousands of concurrent users accessing the service.

TensorLayer: Efficient Deep Learning Development

Developed by Imperial College London with contributions from including Carnegie Mellon University, Stanford University, UCLA, and engineers from Google, Microsoft, Alibaba, Tencent, Bloomberg and others, TensorLayer is described by its developers as a Python-based versatile deep learning library, using MongoDB as the storage layer. In the abstract describing TensorLayer, the authors state "Developing a practical deep learning system is arduous and complex. It involves labor-intensive tasks for constructing sophisticated neural networks, coordinating multiple network models, and managing a large amount of training related data. To facilitate such a development process, we propose TensorLayer which…provides high-level modules that abstract sophisticated operations towards neuron layers, network models, training data and dependent training jobs."

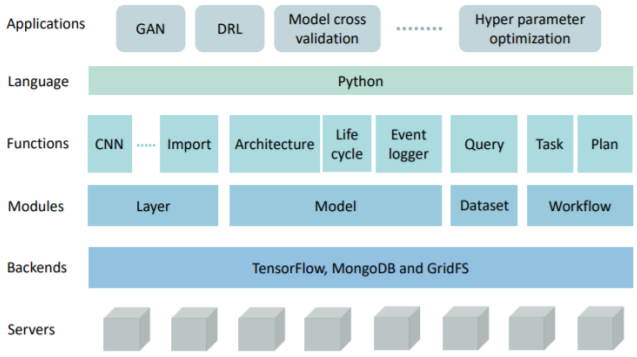

Figure 10: TensorLayer Deep Learning platform

The TensorLayer team selected MongoDB due to the platform's ease of use, performance, rich indexing, GridFS for binary object storage, and availability of management tools.

x.ai: Personal Assistant

x.ai is an AI-powered personal assistant that schedules meetings for its user. Users connect their calendars to x.ai, and then when it's time to set a meeting via email, users instead delegate the scheduling task to 'Amy Ingram' by ccing amy@x.ai. Once she's copied into the email thread, she finds a mutually agreeable time and place and sets up the meeting for you. MongoDB serves as the system of record for the entire x.ai platform, supporting all services

including natural language processing, supervised learning, analytics and email communication. MongoDB's flexible data model has been critical in enabling x.ai to rapidly adapt its training and input data sets, while supporting complex data structures. Learn more by reading the case study.

Auto Trader: Predicting Value

The UK's largest digital car marketplace makes extensive use of machine learning running against data stored in MongoDB. The car's specifications and details, such as number of previous owners, condition, color, mileage, insurance history, upgrades, and more are stored in MongoDB. This data is extracted by machine learning algorithms written by Auto Trader's data science team to generate accurate predictions of value, which are then written back to the database. MongoDB was selected due to its flexible data model and distributed design, allowing scalability across a cluster of more than 40 instances. Learn more from coverage in the technology press.

Mintigo: Predictive Sales & Marketing

Founded by former intelligence agency data scientists, Mintigo delivers a predictive marketing engine for companies such as Red Hat. Through sophisticated machine learning algorithms operating against large data sets stored in MongoDB, Mintigo helps marketing and sales organizations better identify leads most likely to convert to customers. Through its engine, Mintigo users average a 4x improvement in overall marketing funnel efficiency. Mintigo runs on AWS, with machine learning algorithms written in Python. MongoDB is used to store multi-TB data sets, and was selected for scalability of streaming data ingest and storage, and schema flexibility. MongoDB's expressive query framework and secondary indexes feeds the algorithms with relevant data, without needing to scan every record in the database. Learn more from the case study

Geo-Location Analysis for Retail

A US-based mobile app developer has built its Intelligence Engine on MongoDB, processing and storing tens of millions of rich geospatial data points on customers and their locations in real time. The Intelligence Engine uses scalable machine learning and multi-dimensional analytic techniques to surface behavioral patterns that allows retailers to predict and target customers with location-based offers through their mobile devices. MongoDB's support for geospatial data structures with sophisticated indexing and querying provides the foundation for the machine learning algorithms. MongoDB's scale-out design with sharding allows the company to scale from 10s to 100s of millions of customer data points.

Qumram: NLP for Customer Sentiment & Fraud Detection

Qumram's software is used by the most heavily regulated industries in the world to help organizations such as UBS capture every moment of the customer's journey. Every keystroke, every mouse movement, and every button click, across all digital channels, is ingested into MongoDB. From here, Qumram provides a single view of customer interactions, helping its users ensure compliance, prevent fraud, and enrich the online experiences. Qumram uses MongoDB's rich query language to extract insights from the data. It has also turned to Apache Spark with MongoDB for Natural Language Processing (NLP) to extract customer sentiment from their digital interactions, and other deep learning techniques for anti-money laundering initiatives. Learn more from the case study.

Summary

Deep learning and AI have moved well beyond science fiction into the cutting edge of internet and enterprise computing. Access to more computational power in the cloud, advancement of sophisticated algorithms, and the availability of funding are unlocking new possibilities unimaginable just five years ago. But it's the availability of new, rich data sources that is making deep learning real. To advance the state of the art, developers and data scientists need to carefully select the underlying databases that manage the input, training, and results data. MongoDB is already helping teams realize the potential of AI.

Sources

AI, Deep Learning, and Machine Learning: A Primer

Coursera Machine Learning

Neural Networks and Deep Learning

What Machine Learning Can't and Can Do

We Can Help

We are the MongoDB experts. Over 3,000 organizations rely on our commercial products, including startups and more than half of the Fortune 100. We offer software and services to make your life easier:

MongoDB Enterprise Advanced is the best way to run MongoDB in your data center. It's a finely-tuned package of advanced software, support, certifications, and other services designed for the way you do business.

MongoDB Atlas is a database as a service for MongoDB, letting you focus on apps instead of ops. With MongoDB Atlas, you only pay for what you use with a convenient hourly billing model. With the click of a button, you can scale up and down when you need to, with no downtime, full security, and high performance.

MongoDB Stitch is a backend as a service (BaaS), giving developers full access to MongoDB, declarative read/write controls, and integration with their choice of services.

MongoDB Cloud Manager is a cloud-based tool that helps you manage MongoDB on your own infrastructure. With automated provisioning, fine-grained monitoring, and continuous backups, you get a full management suite that reduces operational overhead, while maintaining full control over your databases.

MongoDB Professional helps you manage your deployment and keep it running smoothly. It includes support from MongoDB engineers, as well as access to MongoDB Cloud Manager.

Development Support helps you get up and running quickly. It gives you a complete package of software and services for the early stages of your project.

MongoDB Consulting packages get you to production faster, help you tune performance in production, help you scale, and free you up to focus on your next release.

MongoDB Training helps you become a MongoDB expert, from design to operating mission-critical systems at scale. Whether you're a developer, DBA, or architect, we can make you better at MongoDB.

Resources

For more information, please visit mongodb.com or contact us at sales@mongodb.com.

Case Studies (mongodb.com/customers)
Presentations (mongodb.com/presentations)
Free Online Training (university.mongodb.com)
Webinars and Events (mongodb.com/events)
Documentation (docs.mongodb.com)
MongoDB Enterprise Download (mongodb.com/download)
MongoDB Atlas database as a service for MongoDB (mongodb.com/cloud)
MongoDB Stitch backend as a service (mongodb.com/cloud/stitch)

mongoDB

US 866-237-8815 ▪ INTL +1-650-440-4474 ▪ info@mongodb.com
© 2017 MongoDB, Inc. All rights reserved.