cloudera®

WHITE PAPER

Deep Learning: A Guide for Enterprise Architects

Version: 102

cloudera

Table of Contents

cloudera

Deep learning is in the news.

- It's changing the game.
- It's changing your life.
- It's changing everything.
- It will change the world.

It's good to see people excited about technology. But deep learning is a tool that enterprises use to solve practical problems. Nothing more, and nothing less.

In this paper, we provide an introduction to deep learning and its foundations. Next, we introduce you to Cloudera's unified platform for data and machine learning and show you four ways to implement deep learning.

Finally, we offer six practical tips to help your organization move forward with deep learning.

So you can change your business.

Deep Learning: A Proven Technique

Machine learning is a set of *algorithms* and *methods* that detect useful patterns in data. There are hundreds of different algorithms available to the data scientist; some examples include:

- Linear models
- Decision tree learning
- Kernel-based methods
- Ensemble learning
- Neural networks

Neural networks are a class of machine learning techniques. Developed in the 1940s by neuroscientists to simulate the behavior of human and animal brains, data scientists use them in many different business applications. They are available in open source software libraries and commercial software packages.

A neural network is "deep" if it has specific properties, which we discuss below, under *Deep Learning 101*. "Deep learning" refers to the tools and methods data scientists use to train and deploy deep neural networks. These techniques date to the 1980s; however, application lagged due to the computational complexity and resources needed. Reduced computing costs, the flood of digitized data and improved algorithms make deep learning practical today.

Deep Learning in Action

Deep learning emerged as a useful tool when practitioners used it successfully to win competitions in fields such as document analysis and recognition, traffic sign recognition, medical imaging, and bioinformatics. Today, data scientists use deep learning to a variety of practical problems:

- PayPal, a leading payment systems provider, uses deep learning to detect and prevent fraud.
- Deep Instinct, a startup, uses deep learning to protect against cyber security threats.
- Researchers at Purdue University demonstrate a system that uses deep learning to analyze images and assess disaster damage.
- Lloyds Banking Group uses deep learning to confirm the identity of consumers who call its call center, reducing fraud and improving operations.
- Scientists at Penn State University and the École Polytechnique Fédérale de Lausanne use deep learning to develop a smartphone app that can diagnose disease in plants and crops.
- Zebra Medical Vision, a startup, uses deep learning to diagnose breast cancer.

Deep learning is a proven technique and a key driver for digital transformation. As executives learn more about successful applications, the demand for tools and infrastructure will proliferate.

cloudera

Deep Learning 101

In this section, we offer a brief explanation of neural networks and deep learning. For a more detailed treatment, please refer to the texts linked in the Additional Reading section at the end of this paper.

Data scientists using neural networks specify a problem as a network of nodes, or *neurons*, arranged in *layers*. Directed graphs connect the nodes to one another. The data scientist uses an *optimization* algorithm to find an optimal set of *parameters* for the model, such as weights on edges connecting nodes.

Neurons in an artificial neural network accept data from other neurons as input. They process the data with a mathematical function to produce a calculated result. Data scientists specify the type of function the neuron should apply to incoming data.

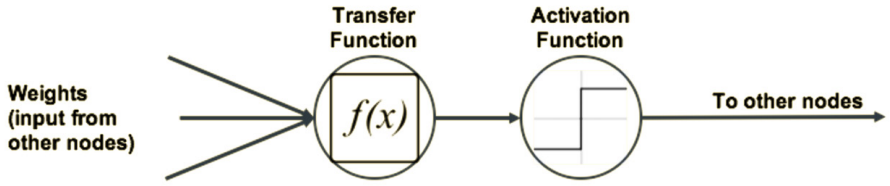

Figure 1: Neural network node

Within the artificial neural network, the data scientist arranges neurons in *layers*. There are three types of layer in an artificial neural network. Neurons in the *input layer* accept data, while neurons in the *output layer* present the result of the model's calculations. The input and output layers of a neural network represent real-world facts: the input layer represents a vector of data, and the output layer represents an object we want to predict, classify, or infer. For example, in an image classification problem, the input is a vector of bit-mapped image data, and the output is a tag that indicates what the image represents — such as "cat."

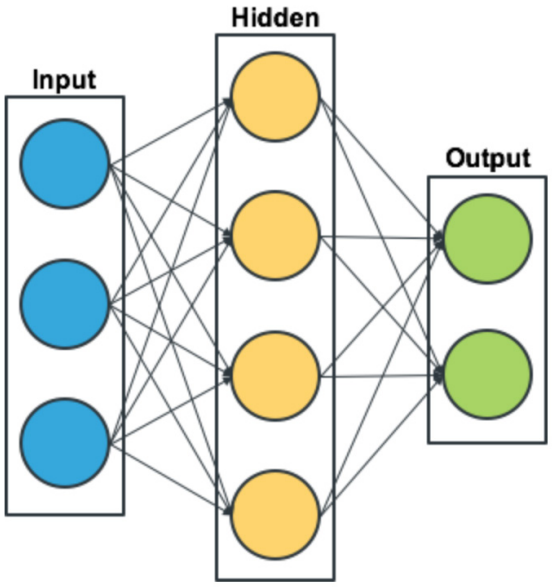

Figure 2: Neural network layers.

Neurons in *hidden layers* perform intermediate calculations. Hidden layers are abstractions that are not directly interpretable; they merely serve to improve the quality of the model. Hidden layers enable neural networks to learn arbitrarily complex functions.

If the artificial neural network has two or more hidden layers, it is a *deep* neural network.

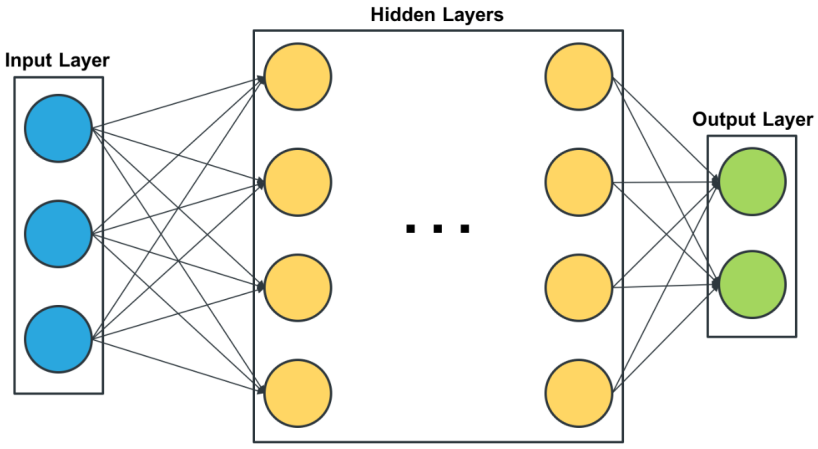

Figure 3: Deep neural network.

Data scientists use the term *architecture* to describe different ways to specify a neural network. There are many different neural network architectures, distinguished by topology, the flow of information, mathematical functions and training methods. Some widely used designs include:

Multilayer Perceptron. Multilayer Perceptrons are a fundamental type of artificial neural network. They are feedforward networks, which means that a neuron in one layer can accept input from neurons in previous layers, but cannot receive information from neurons in the same layer or subsequent layers.

Convolutional Neural Network. Unlike Multilayer Perceptrons, whose neurons are fully connected, neurons in a convolutional network connect locally to neurons in the immediate region to create higher-level features. This arrangement makes them very efficient for image recognition tasks. For example, in image recognition, one neuron represents one pixel in a picture. In a convolutional network, that neuron may connect to neurons that represent surrounding pixels, but not to a neuron that represents a pixel in the far corner of an image.

Recurrent Neural Network. Recurrent networks recognize patterns in sequences of data such as in time series data, handwriting, text, speech, or genomes. Conventional feedforward networks learn from data one case at a time, adjusting weights to minimize errors as they proceed through the data. Recurrent networks, on the other hand, learn, from both the current case and from the state of their output as of the previous case.

There are many other types of neural network, including Radial Basis Function networks, Restricted Boltzmann Machines, Deep Belief Networks, Deep Autoencoders, Recursive Neural Tensor Networks, and Stacked Denoising Autoencoders. For a typology of neural networks, read this article.

Each mathematical function in a neural network has one or more parameters or weights. The number of parameters increases with the size and complexity of the model; in one extreme example, Digital Reasoning, a Cloudera partner, reports training a network for natural language processing with 160 *billion* parameters. A large computational problem requires an efficient optimization algorithm, such as Stochastic Gradient Descent or L-BFGS.

Data scientists train a neural network by running the optimization algorithm with training data. For prediction and inference problems, the training data includes historical examples with a known outcome. The optimization algorithm determines a set of parameters that minimize prediction errors.

Large models require a lot of data. For example, a Microsoft team that completed the ImageNet benchmark used data from 1.3 million images.

Like all machine learning techniques, an artificial neural network delivers business value when the organization applies a trained model to new information. Data scientists call this process *inference*. Inference is the inverse of training. In the training task, the data scientist uses an extensive set of historical examples with known outcomes to estimate the parameters of a model. Inference uses the trained model to predict or infer something that is not known.

Deep Learning Pros and Cons

Deep learning has two key strengths that set it apart from other machine learning techniques. The first of these is *feature learning*. With other techniques, data scientists manually transform features to get the best results with a particular algorithm. This procedure takes time, and also requires a good deal of guesswork. In contrast, deep learning learns higher-level abstractions from input data at many levels. The data scientist does not guess how to combine, recode, or summarize the inputs.

Also, deep learning detects interactions among variables that may be invisible on the surface. It can detect nonlinear interactions and approximate any arbitrary function. While it is possible to fit interaction effects using simpler methods, those methods require manual specification and more guesswork from the data scientist. Deep learning automatically learns these relationships.

Feature learning and the ability to detect complex relationships tend to make deep learning a good choice for certain kinds of data:

High-cardinality outcomes. For problems like speech recognition and image recognition, the learner must distinguish between a huge number of discrete categories. (For example, a speech recognition application must distinguish among almost 200,000 words in English alone.) Mathematicians call this property cardinality. Conventional machine learning techniques often fail at this task; deep learning can solve classification problems with hundreds of thousands of elements.

High-dimension data. In problems such as video analysis, particle physics, or genomic analysis, a data set can have billions of features. Deep learning can work with massively "wide" data sets like that.

Unlabeled data. Labels or tags provide valuable information about a package of data. For example, an image might carry the label "cat." For unsupervised learning, deep learning works with data that lacks informative labels, such as a bit-mapped image.

Deep learning also has some *disadvantages* compared to other machine learning techniques.

Technical challenge. Deep learning is a complicated procedure that requires many choices by the practitioner. These options include such things as network topology, transfer functions, activation functions and the training algorithm. Methods and best practices are nascent; data scientists often rely on trial and error to find a working model. Consequently, deep learning models tend to take much more time than simpler and more mature techniques.

Opaqueness. Deep learning models are difficult or impossible to interpret through inspection of model parameters. Such models may have many hidden layers, which have no "real world" referent. Data scientists evaluate the model by measuring how well it predicts, treating its internal structure as a "black box."

Overfitting. Like many other machine learning techniques, deep learning is prone to overfitting, a tendency to "learn" characteristics of the training data that do not generalize to the population as a whole. Dropout and regularization techniques can help to prevent this problem. As with any machine learning technique, organizations should test and validate the model, and evaluate accuracy with data that is independent of the training data set.

Computationally intensive. Training a deep learning model can require billions of computations. While it is possible to perform this task on conventional hardware, some industry analysts recommend specialized GPU-accelerated platforms. This hardware is not cheap; moreover, due to the demand for high-performance machines, some customers report back orders and extended delivery timelines.

Deployment Issues. Deep learning models are complex, which makes them harder to deploy in a production system. Due to the model's opaqueness, organizations may need to implement additional measures to provide explanations to users.

cloudera

Deep Learning in Cloudera

Cloudera is a unified platform for data and machine learning. With Cloudera, you bring deep learning to your data and not the other way around.

For today's complex technology environments, enterprises need choices and flexibility. Cloudera offers multiple ways to train and deploy deep learning models, without new silos or data movement.

Cloudera Data Science Workbench

Cloudera Data Science Workbench (CDSW), enables fast, easy, and secure self-service data science. It is secure and compliant by default, with support for full Cloudera authentication, authorization, encryption, and governance.

CDSW provides data scientists with a browser-based development environment for Python, R, and Scala. Users can download and experiment with the latest libraries and frameworks in customizable settings, and easily share projects with peers. The software includes built-in scheduling, monitoring, and email alerting.

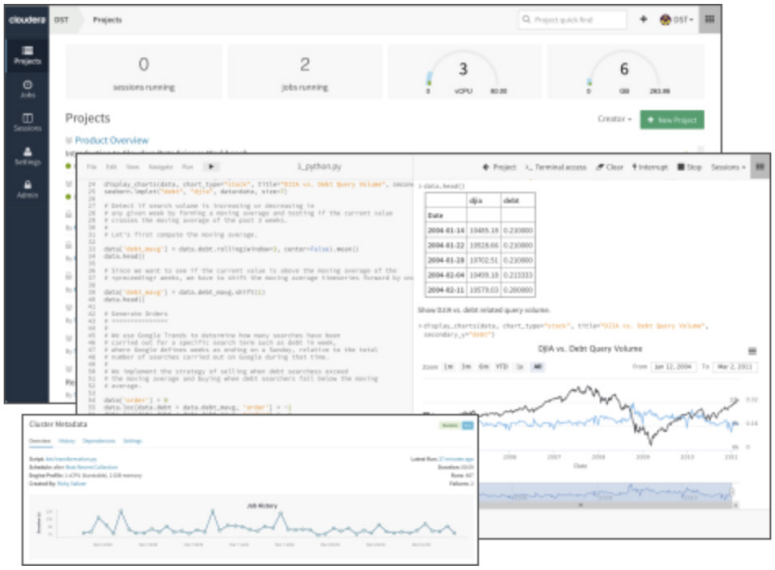

Figure 4: Cloudera Data Science Workbench.

The latest CDSW release includes support for GPU-enabled devices. GPUs are specialized processors that accelerate computationally intensive workloads. GPUs are particularly well suited to the training step for deep learning models. CDSW makes it possible for data scientists to use conventional hardware for tasks like data preparation and discovery, and train a deep learning model on a GPU-accelerated machine.

CDSW users share available GPU resources. Users request a specific number of GPU instances, up to the total number available on a node. CDSW allocates GPUs to the job for the duration of the run. Projects can use isolated versions of libraries, and even different CUDA and cuDNN versions via CDSW's extensible engine feature.

Data scientists working with CDSW can use any deep learning framework that has a Python, R, or Scala API, including TensorFlow, Keras, Theano, Microsoft Cognitive Toolkit (CNTK), Caffe, PyTorch, DL4J, Apache MXNet, Torch, and BigDL.

Learn more about Cloudera Data Science Workbench here.

cloudera®

Apache Spark in Cloudera

Apache Spark in Cloudera provides an excellent platform for transfer learning and inference in an existing cluster. Four open source packages make this possible: BigDL, DL4J, Spark Packages, and Deep Learning Pipelines.

BigDL

BigDL is a distributed deep learning library for Apache Spark, developed and distributed by Intel. With Scala and Python APIs, the software provides broad support for deep learning model development and inference. Also, users can load pre trained TensorFlow, Caffe or Torch models and use BigDL for inference.

BigDL uses the Intel® Math Kernel Library (Intel® MKL) and multithreaded programming in each Spark task. According to Intel, it is orders of magnitude faster than out-of-the-box Caffe, Torch, or Tensor Flow on a single-node Intel® Xeon® processor.

Deeplearning4j

Deeplearning4j (DL4J) is an open source deep learning framework written in Java. Skymind, a Cloudera partner, leads development for the project and provides commercial support.

DL4J is a distributed and multi-threaded framework; it integrates with Spark and trains models within the cluster. In a distributed environment, DL4J shards, or splits large datasets and passes the shards to worker nodes for execution. Each node trains a model on its local data; DL4J then iteratively averages the parameters to produce a single model.

In addition to its Java API, DL4J also supports Scala and Clojure, and a Python interface through Keras.

Spark Packages

Two packages in the Spark Packages library support deep learning:

- TensorFlowOnSpark
- CaffeOnSpark

Yahoo! developed TensorFlowOnSpark and CaffeOnSpark to bring deep learning to Spark clusters. By combining features from the deep learning frameworks and Apache Spark, the packages enable distributed deep learning on clustered servers. They support neural network model training, testing, and feature extraction. Both packages have Python and Scala APIs. Data scientists can combine functions from the deep learning frameworks with other Spark tasks in a single pipeline.

Deep Learning Pipelines

Deep Learning Pipelines is a project of Databricks. It provides high-level APIs for scalable deep learning in Python with Apache Spark. The project, currently in its first release, aims to provide easy-to-use APIs that enable deep learning in very few lines of code. Deep Learning Pipelines supports TensorFlow and TensorFlow-backed Keras workflows. The developers intend to focus on model inference/scoring and transfer learning of image data at scale.

How to Move Forward with Deep Learning

In its most recent Hype Cycle for Data Science and Machine Learning report, Gartner positions deep learning at the Peak of Inflated Expectations:

> *During this phase of overenthusiasm and unrealistic projections, a flurry of well-publicized activity by technology leaders results in some successes, but more failures, as the technology is pushed to its limits. The only enterprises making money are conference organizers and magazine publishers.*

The hype about deep learning poses both opportunity and risk for the enterprise architect. On the one hand, well-publicized success stories raise interest among executives who seek to use deep learning for competitive advantage. On the other hand, excessive enthusiasm can lead the organization to invest in white elephants or drive stakes in the ground that impair its ability to profit from deep learning in the long run.

As with most new technologies, rapidly changing standards make investment challenging. Google released TensorFlow software for deep learning to open source in November 2015; within a few months, it was the most actively developed machine learning project in the open source ecosystem. Since Google's announcement, Amazon, Microsoft, and Intel have all released open source projects for deep learning. While TensorFlow is the most popular deep learning framework used by data scientists today, there is no reason to believe it will retain that status permanently.

In light of deep learning's power and potential, we offer several pragmatic tips to move forward.

Stay focused on solving business problems. Google, Microsoft, and Baidu didn't become deep learning powerhouses because deep learning is cool or because consultants told them that innovation is important. They did so because they had pressing business issues, and deep learning provided a way to solve them.

Deep learning may be the right tool for your organization, too. But if you haven't defined business issues carefully, outlined a strategy to capture and manage data, and tried simpler techniques first, you may build a deep learning capability that nobody uses.

Choose pilot projects carefully. If your organization is new to deep learning, the long-term success of your program may depend on results from your first few projects. Deep learning is most likely to make an impact in projects that include such tasks as:

- Image and video classification or tagging
- Object recognition
- Handwriting recognition
- Speech recognition
- Speech translation
- Text processing

These problems frequently have the properties we identified above as strengths of deep learning: high-cardinality outcomes, dimensionality, and unlabeled data.

Attempts to use deep learning to improve an existing model built with conventional techniques will produce disappointing results most of the time. To produce markedly better results, the data scientist would have to introduce new data to the modeling process. For example, hospitals improve the accuracy of models that predict rehospitalization by adding data from notes to the patient record by medical professionals.

Organize your data first. It may be tempting to let your team dive into training deep learning models. This practice may be useful as a learning opportunity. But keep in mind that behind every deep learning success story there is a data success story.

Successful deep learning applications build on defined data flows for three distinct processes:

- Initial model training
- Updates to the model
- Inference

How well you design these flows will determine the success of your application. For example, while it may be possible to copy a large data set to an offline platform for initial training, it may be costly to do so repeatedly for model updates. In today's fast-paced business, frequent updates to models are the norm in all branches of machine learning. Unless you allow for this, your project may wind up as a high-maintenance "orphan."

Similarly, unless your team thinks through how to run inference with a deep learning model, you may be creating a great model that nobody uses. The business requirements of your application may call for low-latency inference with a service-level guarantee. Build this into your deep learning project plan, or the project will fail.

cloudera®

Embrace open source. Data scientists prefer open source software. All of the most widely used deep learning frameworks are open source. There are some commercial options on the market, but there is no evidence that they outperform the open source frameworks.

Before you standardize on any single framework, consider the results of this benchmark project published in 2016. The authors tested six deep learning problems across five frameworks desktops and servers with and without GPU acceleration. They concluded that no single framework consistently outperforms the others on all problems. The implication is that data scientists should experiment with multiple frameworks to identify the best option for a particular challenge. High-level tools like Keras or Deep Water that work with many deep learning back ends can help.

Leverage transfer learning. Unless your organization already has substantial experience building deep learning models from scratch, pre-trained models are the best way to get started. Check public model repositories, such as the Caffe Model Zoo. If you find a model that approximates the problem you are trying to solve, run it unchanged to establish an accuracy baseline. Use transfer learning to build on existing models instead of starting from scratch. Transfer learning reduces the need for massive training data sets and computing power.

Don't create new silos. Your organization invested millions of dollars and countless hours eliminating silos that impede integration. The last thing you want to do is create a new one. Some vendors argue that deep learning is so new and different that you need an entirely new platform for advanced analytics. Remember: it's a lot harder to bring your data to a deep learning platform than it is to bring deep learning to your data platform.

Cloudera for Deep Learning

Move forward with Cloudera, the unified platform for data and machine learning. You can learn more about Cloudera Data Science and Engineering here.

Thomas Dinsmore, Brad Barker, Seth Hendrickson, Nisha Muktewar, Sean Owen, Vartika Singh, and Tom White contributed to this paper.

Additional Reading

Francois Chollet (2017), Deep Learning with Python, Manning Publications.

Aurélien Géron (2017), Hands-On Machine Learning with Scikit-Learn and TensorFlow, O'Reilly Media.

Ian Goodfellow, Yoshua Bengio and Aaron Courville (2016) Deep Learning, MIT Press.

About Cloudera

Cloudera delivers the modern platform for machine learning and advanced analytics built on the latest open source technologies. The world's leading organizations trust Cloudera to help solve their most challenging business problems by efficiently capturing, storing, processing and analyzing vast amounts of data. Learn more at cloudera.com.

cloudera.com

1-888-789-1488 or 1-650-362-0488
Cloudera, Inc. 395 Page Mill Road, Palo Alto, CA 94306, USA