Deep learning for agronomy

3rd year engineer

Ammouri Bilel

: ESAM
in: ammouri-bilel
: bilelammouri
: Ammouri-Bilel
: 0000-0002-5491-5172

Plan

Keras TensorFlow PyTorch

Sonnet Swift mxnet

DL4J GLUON Caffe

The Microsoft Cognitive Toolkit ONNX

Keras

Keras is an open-source deep learning framework. This tool can operate on top of TensorFlow, Theano, Microsoft Cognitive Toolkit, and PlaidML. Keras' USP is its speed – it comes with built-in support for data parallelism, enabling it to process massive volumes of data while accelerating training time for models. Since it is written in Python, it is incredibly easy to use and extensible.
While Keras excels at high-level computations, low-level computations are not its strong suit. For low-level computations, Keras uses another library called the backend.

Highlights

It is excellent for beginners who are just starting their journey in this field. It facilitates easy learning and prototyping of simple concepts.
It promotes rapid experimentation with deep neural networks.
It aids in writing readable and precise code.

TensorFlow

TensorFlow, Google's leading open-source platform for machine learning and deep learning, is based on JavaScript and provides an extensive set of tools and community resources for the training and deployment of ML/DL models. TensorFlow Lite is used for deploying models on mobile or embedded devices, while TensorFlow Extended is suitable for large-scale production environments. Despite experimental interfaces in JavaScript, C++, C, Java, Go, and Julia, Python remains the preferred programming language. TensorFlow supports model execution on both computing clusters and mobile platforms (iOS and Android) but demands substantial coding and operates with a static computation graph.

Highlights

TensorFlow is best suited for DL model development and experimenting with Deep Learning architectures.
It is used for data integration functions, including combining graphs, SQL tables, and images.

TORCH/PyTorch

PyTorch, an open-source Deep Learning framework developed by Facebook, is based on the Torch library with a primary focus on accelerating the entire process from research prototyping to production deployment. PyTorch is notable for its C++ frontend over a Python interface. The frontend supports model development, while the "torch.distributed"backend facilitates scalable distributed training and performance optimization in both research and production. PyTorch allows the use of standard debuggers like PDB or PyCharm and operates with a dynamically updated graph, enabling architecture changes during the training process.

Highlights

It excels in training, building, and deploying small projects and prototypes.
It is widely used for Deep Learning applications such as natural language processing and computer vision.

Sonnet

Sonnet

Developed by DeepMind, Sonnet serves as a high-level library dedicated to constructing intricate neural network structures within TensorFlow. Positioned as an overlay on TensorFlow, Sonnet's objective is to generate primary Python objects that correspond to specific neural network components. These objects seamlessly integrate with the TensorFlow computation graph, streamlining the creation of high-level architectures. Sonnet revolves around a straightforward yet powerful programming model focused on the "$snt.Module$"concept, emphasizing self-contained and independent modules. While providing predefined modules such as $snt.Linear$, $snt.Conv2D$, $snt.BatchNorm$, and established networks of modules, Sonnet also empowers users to create their own custom modules.

Highlights

Sonnet allows the writing of modules that can declare other sub-modules internally or pass to other modules during the construction process.
Since Sonnet is explicitly designed to work with TensorFlow, it provides easy access to its underlying details, including Tensors and variable$_scopes$.
Models created with Sonnet can be seamlessly integrated with raw TF code and also with those written in other high-level libraries.

Swift

Swift for TensorFlow stands out as a cutting-edge platform that fuses the capabilities of TensorFlow with the Swift programming language, with a specific focus on machine learning applications. Tailored for the field of machine learning, this platform seamlessly incorporates the latest advancements in machine learning, differentiable programming, compilers, system design, and more. While the project is still in its early stages, it welcomes experimentation from anyone interested. In terms of differentiable programming, Swift for TensorFlow leverages first-class auto-diff support, enabling the differentiation of any function or even custom data structures within a matter of minutes. The platform also boasts a sophisticated toolchain designed to enhance user productivity.

Highlights

The powerful Python integration of Swift makes migration extremely easy. By directly integrating with Python, a versatile programming language, Swift for TensorFlow enables users to express powerful algorithms conveniently and seamlessly.
As a statically-typed language, Swift immediately highlights any errors in the code, allowing a proactive approach to fixing them before running the code.

MXNet

MXNet stands as an open-source Deep Learning framework meticulously crafted for the training and deployment of deep neural networks. Renowned for its high scalability, MXNet fosters rapid model learning. Augmenting its appeal is a flexible programming model that accommodates various programming languages, including C++, Python, Julia, Matlab, JavaScript, Go, R, Scala, Perl, and Wolfram. Emphasizing adaptability, MXNet is portable and seamlessly adjusts to multiple GPUs and diverse machine configurations. This lightweight, flexible, and scalable deep learning framework is well-equipped to support cutting-edge models, such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks.

Highlights

It supports multiple GPUs with fast context switching and optimized computation.
It supports both imperative and symbolic programming, allowing developers to choose the programming approach they prefer for building deep learning models.

DL4J

Deeplearning4J (DL4J) is a distributed Deep Learning library written for Java and the Java Virtual Machine (JVM). It is compatible with all JVM languages such as Scala, Clojure, and Kotlin. In DL4J, underlying computations are written in C, C++, and CUDA.
The platform utilizes both Apache Spark and Hadoop, accelerating model learning and integrating AI into professional environments for use on distributed CPUs and GPUs.
It is powered by its unique open-source numerical computing library, ND4J.
In ND4J, neural networks are trained in parallel through iterative reduction across clusters.
DL4J incorporates implementations of Restricted Boltzmann Machines, Deep Belief Networks, Deep Autoencoder, Recursive Neural Tensor Network, Stacked Denoising Autoencoder, word2vec, doc2vec, and GloVe.

Highlights

With DL4J, it is possible to compose deep neural networks from shallow networks, each forming a "layer."This flexibility allows users to combine variational autoencoders, sequence-to-sequence autoencoders, convolutional networks, or recurrent networks as needed in a distributed, production-quality framework that works with Spark and Hadoop.

Gluon

Gluon is an open-source Deep Learning interface that assists developers in easily and quickly building machine learning models. It provides a simple and concise API for defining ML/DL models using a variety of pre-built and optimized neural network components. Gluon enables users to define neural networks using simple, clear, and concise code. It comes with a comprehensive range of ready-to-use neural network building blocks, including predefined layers, optimizers, and initializers. These components help eliminate much of the underlying complex implementation details.
It is based on MXNet and provides a neat API that simplifies DL model creation.
Gluon juxtaposes the training algorithm and the neural network model, providing flexibility in the development process without compromising performance. This training approach is known as the Gluon training method.
Gluon allows users to opt for a dynamic definition of the neural network, meaning it can be built on the fly using the structure of their choice and Python's native control flow.

Highlights

Since Gluon allows users to define and manipulate ML/DL models like any other data structure, it serves as a versatile tool for beginners entering machine learning.
Thanks to Gluon's high flexibility quotient, it is easy to prototype and experiment with neural network models.

ONNX

The Open Neural Network Exchange (ONNX) project is a collaboration between Microsoft and Facebook. It is an open ecosystem designed for the development and deployment of ML and DL models. It includes the definition of an extensible computation graph model and standard operator definitions and data types. ONNX simplifies the process of transferring models between different means of working with AI – you can train models in one framework and transfer them to another for inference.
ONNX was designed as an intelligent system for switching between different ML frameworks such as PyTorch and Caffe2.
ONNX models are currently supported in Caffe2, Microsoft Cognitive Toolkit, MXNet, and PyTorch. You will also find connectors for several other standard libraries and frameworks.

Highlights

With ONNX, accessing hardware optimizations becomes easier. ONNX allows users to develop in their preferred framework with the chosen inference engine, without worrying about downstream inference implications.

Caffe

Recognized for its exceptional speed, Caffe stands out as a high-performance deep learning framework with support for interfaces such as C, C++, Python, MATLAB, and Command Line. Particularly renowned for its efficacy in modeling Convolution Neural Networks (CNN), Caffe has gained widespread popularity in recent years.
One of the standout features of Caffe's C++ library is its seamless access to the 'Caffe Model Zoo.' This repository houses pre-trained networks, offering immediate usability for various applications, whether it involves modeling CNNs or addressing image processing challenges. Caffe emerges as the preferred library for these tasks.
The primary strength of Caffe lies in its remarkable speed, capable of processing over sixty million images daily with a single Nvidia K40 GPU.

Highlights

C++ library comes with a Python interface.
The configuration defines models without hard-coding.
Easier to set up and train, without having to build onto the network.
Support for recurrent neural networks is quite poor.

Microsoft Cognitive Toolkit (CNTK)

The CNTK is widely acclaimed for its user-friendly training process and the versatile combination of popular model types across servers. This open-source deep learning framework is specifically designed for training deep learning models, excelling in the efficient implementation of Convolutional Neural Networks (CNNs) and training for image, speech, and text-based data.

A notable strength of the CNTK lies in its seamless resource utilization, facilitating the quick implementation of Reinforcement Learning models or Generative Adversarial Networks (GANs). Compared to toolkits like Theano or TensorFlow, it stands out for providing superior performance and scalability, especially when operating across multiple machines.

One of the toolkit's distinctive features is its support for inventing new, complex layer types without requiring users to implement them in a low-level language.

Highlights

Highly efficient and scalable for multiple machines.
Supported by interfaces such as Python, C++, and Command Line.
Fit for image, handwriting and speech recognition use cases.
Supports both RNN and CNN type of neural networks