TensorFlow - Biblioteca para Machine Learning

Gisela Miranda Difini, Leonardo Felix

Curso de Bacharelado em Ciência da Computação - Universidade do Vale do Rio dos Sinos (UNISINOS) - São Leopoldo
93020-190 – São Leopoldo – RS– Brasil

giselamd@edu.unisinos.br, leonardofelix@edu.unisinos.br

Abstract. *TensorFlow is an interface that communicates with the operating system, giving complete control to the developer to manage resources and processes of machine learning with efficiency. This open-source library, or API, makes it faster to build a machine learning environment that can be used in many different systems, varying from smartphones to large-scale distributed systems. This library is widely used for many purposes, for research and development in fields like speech recognition, robotics, data analyses, and many more.*

Resumo. *TensorFlow é uma interface que permite a comunicação com o sistema operacional, dando controle ao desenvolvedor, de forma facilitada, mantendo sob sua responsabilidade o gerenciamento de recursos, visando aumentar a eficiência no processo de aprendizado de máquina. Esta biblioteca de código aberto, ou API, agiliza a otimização dos processos de aprendizado de máquina e pode ser utilizada em diversos sistemas uniformemente, podendo variar de dispositivos móveis a computadores pessoais ou até mesmo sistemas de larga-escala trabalhando de forma paralela. Esta biblioteca é utilizada amplamente, em pesquisas e sistemas de produção, ao longo de diversas áreas de tecnologia, como o reconhecimento de falas, robótica e análise de dados.*

1. Introdução

De acordo com Walker, J. (Gartner, 2017), tecnologias de inteligência artificial serão as mais disruptivas nos próximos 10 anos, devido ao poder de computação sem precedentes com quantidade de dados quase infinitas e o avanços nas áreas de redes neurais, porém a análise de toda essa carga de informação também se torna difícil sem modelos de aprendizado de máquina eficientes. Como acelerador deste processo, surge o TensorFlow, uma biblioteca de software de código-fonte aberto para Machine Intelligence.

A história do TensorFlow começa em 2011, com o projeto da Google denominado de Google Brain. Para este projeto foi criado a primeira geração de bibliotecas para aprendizado de máquinas, o DistBelief. De acordo com Abadi et. al. (2015), após entenderem melhor as necessidades e os requisitos para o processo de aprendizado de máquina, surgiu a necessidade de uma interface mais flexível, escalável e com melhor performance, o TensorFlow, sendo este a segunda geração da biblioteca de código aberto para aprendizado de máquina, otimizada.

2. TensorFlow

Ao contrário do DistBelief, TensorFlow é mais inteligente, mais flexível e mais veloz que o seu sistema anterior, e muito mais escalonável devido a sua abstração de programação baseada em fluxo de dados que permite aos seus usuários a sua utilização tanto em sistemas pequenos, como *smartphones*, quanto em larga escala, em sistemas distribuídos com milhares de computadores.

O TensorFlow utiliza a representação de fluxo de dados para seus modelos, sendo cada operação matemática representada como nós no gráfico de fluxo de dados (Figura 1). Esse gráfico expressa a comunicação entre sub computações explicitamente, assim facilitando a execução de cálculos independentes em paralelo e particionamento de cálculos em vários dispositivos. Construir camadas de operadores simples facilita a diferenciação destes modelos automaticamente. Sua interface de script de alto nível envolve a construção de gráficos de fluxo de dados permitindo que os usuários experimentem diferentes arquiteturas e algoritmos de otimização sem modificar o núcleo do sistema.

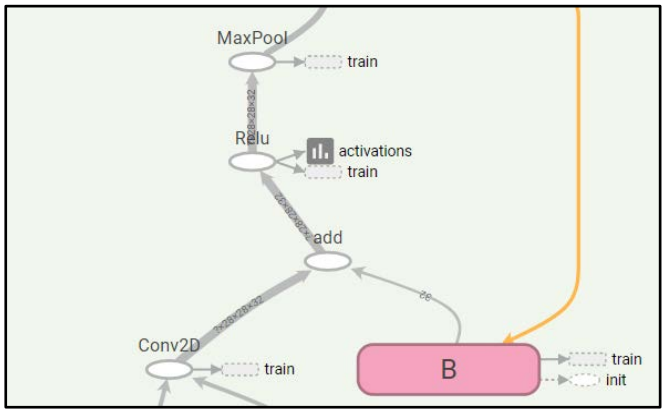

Figura 1. Representação de fluxo de dados no TensorFlow.

Fonte: Os Autores

Para suportar aceleradores voltados ao *Deep Learning* em TensorFlow, definiu-se uma abstração comum para dispositivos. No mínimo, um dispositivo deve implementar métodos para: emissão de um kernel para execução, alocação memória para entradas e saídas, e transferência de buffers da memória e para a memória do host. Cada operador (por exemplo, multiplicação de matriz) pode ter múltiplas implementações especializadas para diferentes dispositivos. Sendo assim, o programa pode segmentar facilmente GPUs, TPUs (*Tensor Processing Unit*) ou CPUs móveis conforme necessário.

TensorFlow utiliza seus tensores com valores primitivos para que todos os dispositivos entendam. Esta decisão garante que os níveis mais baixos do sistema tenham implementações simples para alocação de memória e serialização, reduzindo assim a sobrecarga da estrutura. Os tensores também permitem outras otimizações de gerenciamento de memória e de comunicação, como RDMA (Remote Direct Memory Access), ou seja, sem envolver nenhum dos sistemas operacionais e transferência direta de GPU para GPU.

3. Comunicação com o sistema operacional

Com a grande quantidade de recursos necessários para a implementação de algoritmos de aprendizado de máquina, a implementação em sistemas centralizados se torna

complexa e o gerenciamento destes recursos de forma manual se torna inviável, de acordo com a escala necessária para a execução de tais em tempo viável. Desta forma, a biblioteca TensorFlow consegue fazer o gerenciamento de filas, de sessões e da distribuição do modelo de aprendizado de máquina. Devido à sua implementação flexível, o TensorFlow permite que os dados ou instruções sejam passados entre dispositivos com diferentes sistemas operacionais ou funcionalidades, como de uma CPU de um dispositivo móvel, para a TPU ligada em um servidor, por exemplo, e a coleta dos resultados após o processamento, com a utilização de apenas algumas linhas de comando. Para colocar as operações em um processo específico, pode-se usar a função *tf.device* que é usada para especificar se as operações serão executados na CPU ou na GPU, como exemplificado na Figura 2.

```
with tf.device('/gpu:0'): #GPU:0 calcula a^n e armazena em c2
    a = tf.placeholder(tf.float32, [10000, 10000])
    c2.append(matpow(a, n))

with tf.device('/gpu:1'): #GPU:1 calcula b^n e armazena em c2
    b = tf.placeholder(tf.float32, [10000, 10000])
    c2.append(matpow(b, n))

with tf.device('/cpu:0'): #a CPU faz a adição de todos os valores
  sum = tf.add_n(c2)
#roda o algoritmo, passando o vetor A para a, e o vetor B para b
with tf.Session() as sess:
    sess.run(sum, {a:A, b:B})
```

Figura 2. Exemplo de execução em paralelo no TensorFlow.

Fonte: Os Autores

O TensorFlow é uma biblioteca de várias plataformas. Uma API em C separa o código do nível do usuário em diferentes linguagens do tempo de execução do núcleo (Figura 3). Sua biblioteca principal é implementada em C++ e Python para portabilidade e desempenho, podendo ser executada em vários sistemas incluindo Linux, Mac OS X, Windows, Android, e iOS [Abadi et. al. 2016]. Seus principais componentes são o cliente, que usa a interface de sessão para se comunicar com o mestre e um ou mais processos de trabalho, com cada processo de trabalho responsável por arbitrar o acesso a um ou mais dispositivos computacionais (como núcleos de CPU ou cartões de GPU) e para executar nós de gráficos nesses dispositivos conforme instruído pelo mestre [Abadi et. al. 2015]. As implementações da interface TensorFlow podem ser locais e distribuídas, possibilitando a comunicação com vários processos em máquinas diferentes. Também é possível realizar execuções em paralelo ou em etapas simultâneas, concorrentes.

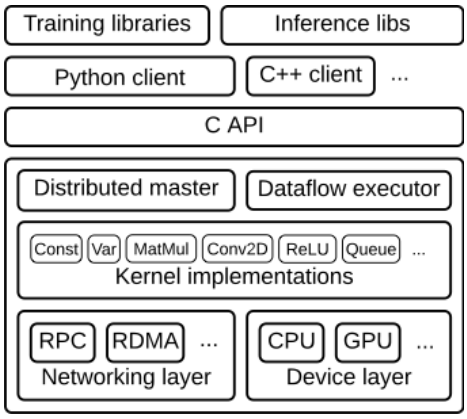

Figura 3. Arquitetura geral do TensorFlow.

Fonte: Tensorflow.org

4. Conclusão

Com a crescente geração de informações, o TensorFlow auxilia no processamento de dados de forma significativa, algo que se tornaria muito mais difícil sem uma ferramenta unificada para gerenciar o aprendizado de máquina, independente da plataforma. TensorFlow é um sistema de machine learning baseado em redes neurais de deep learning, ou seja, um sistema que cria uma rede neural artificial que pode aprender e realizar decisões inteligentes por conta própria. Ter um sistema único para essa finalidade é necessário para aplicar aprendizado de máquina, evitando manutenção excessiva e sistemas defeituosos. Ademais, sua flexibilidade em relação à sua implementação traz muita vantagem aos desenvolvedores.

Referências

Martín Abadi, Ashish Agarwal et. al. “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems”. Disponível: https://arxiv.org/pdf/1603.04467.pdf, Novembro/2015. Acesso em: outubro/2017.

Giancarlo Zaccone “Getting Started with TensorFlow”. Disponível: https://books.google.com.br/books?hl=pt-BR&lr=&id=rsyqDQAAQBAJ&oi=fnd&pg=PP1&dq=tensorflow&ots=7O1V04Q5us&sig=J2ki8GknTPq-hxLlVp-up6gOd9U#v=onepage&q&f=false, Julho/2016. Acesso em: outubro/2017.

TensorFlow Official Documentation “TensorFlow”. Disponível: https://www.tensorflow.org, Novembro/2017. Acesso em: novembro/2017.

Panneta, Kasey “Top Trends in the Gartner Hype Cycle for Emerging Technologies, 2017”. Disponível: https://www.gartner.com/smarterwithgartner/top-trends-in-the-gartner-hype-cycle-for-emerging-technologies-2017/, agosto/2017. Acesso em: novembro/2017.

Martín Abadi, Ashish Agarwal et. al. “TensorFlow: A System for Large-Scale Machine Learning”. Disponível: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf, Novembro 2016. Acesso em: novembro/2017.