Estudo comparativo de plataformas de Deep Learning: Apache Singa, Graphlab e H2O

Elias Augusto Fank[1], **Geomar A. Schreiner**[1,2], **Denio Duarte**[1]

[1]Universidade Federal da Fronteira Sul
Campus Chapecó
Chapecó – SC – Brasil

[2]Universidade Federal de Santa Catarina
Florianópolis – SC – Brasil

eliasfank@hotmail.com, schreiner.geomar@uffs.edu.br, duarte@uffs.edu.br

Abstract. *Deep learning techniques have been showing advances in various machine learning tasks. However, the implementation of these techniques is very complicated. Thus, to help with the implementation of Deep Learning projects, software platforms have been proposed. A considerable number of these platforms already exist, leading to difficulty choosing which platform is the better option to start a deep learning project. This work proposes a comparative study between some well-established platforms for Deep Learning projects: Apache Singa, Graphlab, and H20. This study aims to help users in choosing a platform. We conduct some experiments using two datasets: MNIST and KDD Cup 1999. Results show that the tested platforms are suitable for Deep Learning projects.*

Resumo. *Técnicas de Deep learning vêm mostrando avanços em várias tarefas de aprendizado de máquina. Porém a implementação dessas técnicas é muito complexa. Assim, para ajudar na implementação de projetos de Deep Learning, plataformas estão sendo criados. Já existe uma quantidade considerável destas plataformas disponível. Isso acaba trazendo uma dificuldade na escolha de quem procura começar um projeto. Com o objetivo de auxiliar nesta escolha, este trabalho faz um estudo comparativo entre algumas plataformas: Apache Singa, Graphlab e H2O. Experimentos são conduzidos utilizando os conjunto de dados MNIST e KDD Cup 1999. Resultados apontam que as plataformas testadas têm suas vantagens: Graphlab é a mais intuitiva, a Apache Singa oferece mais recursos e H2O obteve os melhores resultados de predição.*

1. Introdução

Inúmeros problemas podem ser resolvidos e explorados a partir de sistemas de aprendizado de máquina (AM), como, por exemplo, a identificação de objetos em imagens, transformação de trechos de fala em texto e recomendação de produtos aos consumidores de *sites e-commerce* baseando-se em suas buscas.

Segundo [Duarte and Ståhl 2019], aprendizado de máquina é uma área da computação que busca melhorar a autonomia dos computadores através da sua própria experiência. Ou seja, projetar algoritmos que possam aprender a partir de um conjunto de dados de entrada. Levando em consideração a complexidade de tarefas que podem ser

resolvidas com o uso de AM, algumas das técnicas convencionais vêm sendo substituídas por algoritmos baseados em *Deep Learning*. Assim, [LeCun et al. 2015] apontam que quando um problema é complexo, os dados disponíveis costumam estar representados de uma forma também complexa, e com algoritmos de *Deep Learning* é possível, automaticamente, extrair e descobrir qual é a melhor representação para cada característica dos dados. Desta forma, elimina-se uma etapa inicial que é transformar os dados de entrada em estruturas prédefinidas para que os algoritmos de AM possam funcionar corretamente.

Deep learning não é uma técnica recente, pois já faz parte de várias aplicações na atualidade. Sua utilização vem se intensificando por consequência do aumento do poder computacional das máquinas (principalmente do processamento gerado por placas gráficas cada vez mais robustas), da popularização das técnicas para criação de modelos baseados em *Deep Learning* e do tamanho dos dados utilizados para treinamento [Deng and Yu 2014].

O uso crescente das técnicas baseadas em *Deep Learning* fez com que plataformas fossem propostas para auxiliar no desenvolvimento de sistemas com resoluções mais adequadas. Estas plataformas permitem um desenvolvimento e uma implementação mais eficiente [Shatnawi et al. 2018]. Como exemplos de plataformas que facilitam a criação de projetos de *Deep Learning* podem-se citar: *Singa*, *H2O* e *GraphLab*, as quais serão objetos de estudo neste trabalho.

Neste contexto, o objetivo deste trabalho é comparar plataformas que ofereçam suporte para o desenvolvimento de modelos de *Deep Learning*, identificando as suas características em termos de desempenho computacional (utilização de recursos, tempos de treinamento e execução) e dificuldade de implementação. Resultados apontam que as plataformas testadas têm suas vantagens: (*i*) *Graphlab* é a plataforma mais intuitiva de configurar e testar, (*ii*) *Apache Singa* oferece mais recursos porém obteve obteve o pior desempenho geral, e (*iii*) *H2O* obteve os melhores resultados de predição.

O restante deste trabalho está organizado da seguinte forma: a próxima seção apresenta as plataformas objetos da comparação, a Seção 3 apresenta os trabalhos relacionados. Na Seção 4 são apresentados os experimentos e, finalmente, a Seção 5 conclui este trabalho.

2. Plataformas

Nesta seção são apresentadas brevemente as três plataformas de *Deep Learning* comparadas neste trabalho. Para fins de comparação, são elencados os principais pontos de funcionamento de cada uma das plataformas, e quais os algoritmos tradicionais de *Deep Learning* disponibilizados por elas. São considerados como algoritmos tradicionais aqueles baseados em redes neurais (RN) [Guo et al. 2016], como, por exemplo: (*i*) *Convolutional Neural Networks* (CNN) que correspondem às RN propícias às tarefas que envolvam imagens, (*ii*) *Recurrent Neural Network* (RNN) que são utilizadas em reconhecimento de textos e discursos, (*iii*) *Restricted Boltzmann Machine* (RBM) que possui uma utilização mais genérica (*e.g.*, redução de dimensionalidade, classificação e seleção de atributos), (*iv*) *Deep Belief Networks* (DBN) que são especializações das RBMs, e (*v*) *Multilayer Perceptrons* (MLP) que é a mais tradicional e menos complexa em relação às anteriores.

O *Apache Singa*[1] é uma plataforma de *Deep learning* distribuída que traz modelos como redes neurais convolutivas (CNN), máquinas de Boltzmann restritas (RBM) e redes neurais recorrentes (RNN) [Ooi et al. 2015]. Essa plataforma suporta *frameworks* para treinamento síncrono, assíncrono e híbrido. O treinamento síncrono melhora a eficiência de uma iteração, e o treinamento assíncrono melhora a taxa de convergência. Usuários desta plataforma também podem escolher rodar um *framework* híbrido para ter um equilíbrio entre eficiência e taxa de convergência.

Além disso, o *Apache Singa* é um software de código aberto disponível sob a licença "Apache 2.0". A plataforma possui dois componentes principais: *Worker & Server* e *TrainOneBatch*. *Worker & Server* são arquiteturas lógicas criadas para organização e que facilitam a distribuição dos processos. Um *Worker* executa as atualizações dos parâmetros da rede, e um *Server* mantém as informações atualizadas e as distribui para cada *Worker*. *TrainOneBatch* é uma função evocada pelo *Worker* em cada iteração do algoritmo SGD (algoritmo gradiente descendente). Essa função executa basicamente duas tarefas: a de *back-propagation* (BP) e a de *contrastive divergence* (CD).

Outra plataforma escolhida é o *H2O*[2]. Diferentemente da plataforma *Singa*, vem com um propósito mais geral para AM. O *H2O* também é disponibilizado sob a licença "Apache 2.0", e possui um código *Open Source*. Segundo [Candel et al. 2016], utilizando compressão de memória, o *H2O* consegue trabalhar com um grande volume de dados na memória mesmo em um *cluster* de pequena escala. Possui também interface de acesso compatível com várias linguagens de programação como *R*, *Python*, *Scala*, *Java* e *CoffeeScript/JavaScript*.

Um diferencial do *H2O* é o suporte a uma estrutura *multi-cluster*. Sendo assim, no modo *Deep learning*, a plataforma é capaz de distribuir tarefas em vários nós. Para operar neste modo, inicialmente, os dados de treinamento são distribuídos para cada nó. Após a distribuição, a cada iteração, o nó faz uma cópia dos parâmetros globais (informações dos outros nós) e computa os parâmetros de sua rede individualmente.

A terceira plataforma analisada foi a plataforma *GraphLab*. Dentre as plataformas analisadas, esta é a de propósito mais geral sendo utilizada para tarefas de AM ou para visualização e análise de dados [Low et al. 2012]. Foi criada sob uma estrutura implementada na linguagem C++ e possui biblioteca disponível em Python. Possui uma estrutura formada basicamente por cinco componentes lógicos. O primeiro, *Data graph*, é utilizado para guardar o estado do programa, como informações do usuário e o estado da execução atual. O segundo componente é o *Update function* que é um procedimento sem estados que modifica os dados. O próximo componente, *Scheduling primitives*, tem a função de manter a ordenação da execução da ferramenta. O quarto componente é o *Data consistency model*, que é responsável por impedir que o cálculo de cada função independente sobreponha os dados de outras (fator que mantém o paralelismo consistente). O último componente é o *Sync mechanism*, responsável pela sincronização das tarefas. Utilizado, principalmente, para a opção distribuída, tanto em várias GPUs, quanto no caso de *multi-cluster*.

Considerando as plataformas apresentadas é proposta a Tabela 1 que sumariza

[1] `http://singa.apache.org`
[2] `http://www.h2o.ai/`

Plataforma	Redes implementadas	API disponível	Interface
Singa	MLP, CNN, RBM, RNN	Python e C++	Linha de comando
H2O	MLP, CNN*, RNN*	Python, R, Java e Scala	UI
Graphlab	MLP, CNN	Python	UI

Tabela 1. Comparativo das plataformas *Graphlab*, *H2O* e *Apache Singa*

algumas características de cada plataforma. Através dela é possível identificar quais são as redes em comum que estão presentes tanto em *Graphlab*, *H2O* e *Apache Singa*. As redes marcadas com "*" estão disponíveis nas respectivas plataformas apenas com uso de bibliotecas de terceiros, por isso, a rede em comum que será utilizada neste trabalho é a MLP pois está presente, nativamente, em todas as plataformas. As plataformas permitem o uso de *Python* como API de programação, assim os experimentos são implementados nessa linguagem de programação.

3. Trabalhos Relacionados

Os trabalhos selecionados para discussão são aqueles que realizam comparações entre *frameworks* ou plataformas. Dentro deste contexto, foram selecionaods 3 trabalhos apresentados brevemente a seguir.

A comparação proposta em [Bahrampour et al. 2015] tem como objetivo principal comparar os *frameworks*: *Caffe*, *Neon*, *TensorFlow*, *Theano* e *Torch*. O estudo compara a utilização de *hardware* e velocidade de execução dos algoritmos. Esses resultados são de dados coletados em testes executados em *frameworks* que foram executados em uma máquina (*single cluster*) tanto na configuração para CPU quando para GPU.

Os resultados da comparação são: (*i*) *Theano* e *Torch* são de uso mais genérico e suportam diferentes arquiteturas de *Deep learning*, (*ii*) no treinamento e execução das redes na CPU, *Torch* foi o melhor, seguido por *Theano*, (*iii*) na execução de redes convolutivas e *fully-connected* na GPU, *Torch* foi o melhor seguido por *Theano*, (*iv*) no treinamento em uma rede convolutiva e *fully-connected* na GPU, foi percebido que *Theano* é o mais rápido para pequenas redes e *Torch* é o mais rápido para redes grandes, e (*v*) a *Neon* tem grande potencial para grandes redes convolutivas treinadas utilizando GPU.

Em [Kovalev et al. 2016] é apresentado um estudo comparativo entre os seguintes *frameworks*: *Theano*, *Torch*, *Caffe*, *TensorFlow* e *DeepLearning4J*. São feitas avaliações quantitativa dos tempos de treinamento e das predições desses *frameworks*. Diferentemente do trabalho anterior, este concentra-se apenas em fazer o estudo para redes com arquitetura *fully-connected*, variando a largura e a profundidade da rede. Também faz comparativos quando varia-se a função de ativação (*relu* versus *tanh*). Foram utilizadas quatro métricas de avaliação: tempo de convergência, tempo de predição ou de classificação, acurácia na classificação, e a complexidade da solução desenvolvida. Considerando as métricas propostas pelos autores, os resultados foram em ordem de desempenho geral: *Theano* (utilizando *Keras*), *Tensorflow*, *Caffe*, *Torch*, *Deeplearning4J*.

O terceiro e último trabalho proposto por [Ng et al. 2016] apresenta uma avaliação para as plataformas *Singa* e *H2O* comparando a acurácia e o tempo de treinamento. Em sua conclusão, esse trabalho mostra que a plataforma *H2O* obteve uma performance estável e bem acurada para a base de dados *MNIST*. E por outro lado, *Singa* mostrou

Trabalho	Ferramentas Comparadas
[Bahrampour et al. 2015]	Caffe, Neon, TensorFlow, Theano e Torch
[Kovalev et al. 2016]	Theano, Torch, Caffe, TensorFlow e Deeplearning4J
[Ng et al. 2016]	Apache Singa e H2O
Proposta	Apache Singa, Graphlab e H2O

Tabela 2. Características dos trabalhos relacionados e a proposta

ter um bom desempenho quando a rede é treinada em um curto período de tempo, embora a acurácia varie bastante e além do esperado quando os parâmetros de treinamento são alterados. Os autores indicam que para problemas que tenham restrição de tempo de treinamento, *Singa* tem o melhor desempenho. Caso contrário, *H2O* tem um melhor desempenho geral. Outro ponto de destaque é que a implementação da plataforma *Singa* possui uma utilização menos intuitiva, quando se trata dos detalhes de treinamento, de modo que os usuários precisam configurar os parâmetros com mais cautela ao criar um projeto.

Como pode-se observar na Tabela 2, os trabalhos de [Bahrampour et al. 2015] e [Kovalev et al. 2016] têm seu foco na comparação de *frameworks*. Assim, o trabalho relacionado mais próximo ao presente trabalho é o de [Ng et al. 2016] que faz a comparação de duas plataformas de *Deep learning*. A proposta aqui apresentada, além de incluir *Graphlab*, analisará também a complexidade (linha de código), uso de memória e tempo de predição.

4. Experimentos

Esta seção apresenta os experimentos realizados para a avaliação das três plataformas: *Graphlab*, *H2O* e *Singa*. Inicialmente, são apresentadas as bases de dados utilizadas nos testes, então a configuração do ambiente de execução, as métricas a serem analisadas e por fim os experimentos de avaliação das plataformas.

4.1. Conjunto de dados

Para obter resultados mais generalizados, optou-se por fazer a utilização de dois conjuntos de dados com princípios diferentes. Em um deles, cada exemplo é composto por uma imagem com o respectivo rótulo. No outro, os exemplos são compostos por atributos numéricos e textuais mais o rótulo de cada exemplo. Ambos os conjuntos de dados podem ser utilizadas, então, para treinamentos do tipo supervisionado. Estes conjuntos são apresentados a seguir.

MNIST[3]: é um conjunto de dados que armazena dígitos escritos a mão disponíveis no formato de imagem. MNIST é composta por 60 mil exemplos para treinamento e 10 mil exemplos para teste. Existem 10 possíveis rótulos que representam os dígitos de 0 a 9. Este conjunto de dados está disponível gratuitamente para *download* no formato binário. Para facilitar o uso do MNIST nas ferramentas, os dados de treinamento e teste foram convertidos para o formato CSV. Assim cada exemplo, que é formado por uma imagem de 28x28 pixels e seu respectivo rótulo, gerou uma linha no arquivo CSV com 784 colunas

[3]http://yann.lecun.com/exdb/mnist/

(uma para cada pixel 28x28) mais uma coluna para o rótulo. Os tamanhos dos arquivos em *Mb* foram de 109.5 e 18.2 para o treinamento e teste, respectivamente.

KDD Cup 1999[4]: diferentemente do MNIST, este conjunto de dados não possui dados que representam imagens. Foi utilizado em uma competição para construir um detector de intrusão de rede baseado em *logs* de acesso. Um modelo preditivo capaz de distinguir entre conexões "ruins", chamadas intrusões ou ataques, e conexões normais "boas". O conjunto de dados possui 4.898.431 instâncias com aproximadamente 743 *Mb*. Além do conjunto rotulado para testes, com 311.029 instâncias e 47.25 *Mb*.

4.2. Métricas

Com o objetivo de obter resultados comparáveis entre as plataformas *Graphlab*, *H2O* e *Singa* foram definidas quatro métricas: (*i*) acurácia que indica o percentual de acertos das classes dos conjuntos de dados, (*ii*) tempo de treinamento de cada plataforma (em segundos), (*iii*) tempo de predição, *i.e.*, tempo (em segundos) para a plataforma predizer exemplos não vistos (foi utilizado o pacote *time* da linguagem *Python*), (*iv*) consumo de memória em que é feito o registro do pico de memória consumida, em MB, pela plataforma durante a execução do treinamento (foi utilizada a ferramenta *time* do *Linux*), e (*v*) número de linhas para desenvolver a solução.

Para validar as comparações entre as plataformas, foi utilizado o teste *Student's T-Test* (*t-test*) [Zimmerman 1997]. Esse teste é feito sobre dois conjuntos de dados e o seu resultado é um número entre 0 e 1 que mede a confiança de uma afirmação. Neste trabalho, as afirmações que são passíveis de validação são de que uma plataforma obteve um melhor resultado do que a outra em determinado teste. Assim, criou-se duas hipóteses que serão levantadas nas seções dos experimentos. Na hipótese H_1 que indica que as duas plataformas comparadas são iguais caso o resultado do *t-test* for $\alpha > 0,05$ (95% de confiança). E na hipótese H_2 pode-se afirmar que uma plataforma foi melhor ou pior que a outra em um determinado aspecto quando $\alpha \leq 0,05$

4.3. Configuração do ambiente de testes

Para a execução dos experimentos, foi utilizada uma única máquina com sistema operacional Ubuntu Linux 16.04 (64 bits). Essa máquina possuía 8GB de memória *RAM*, um processador *Intel Core i5 - 3470 @ 3,20 GHz* e um *HD* de 1TB - 7200 *RPM*. Nesta máquina foram instaladas as plataformas *Graphlab* (*v2.1*), *H2O* (*v3.18.0.11*) e *Singa* (*v1.2.0*) seguindo a documentação disponível. Todos os algoritmos criados nas três plataformas foram desenvolvidos na linguagem de programação Python (*Python 2.7.0*). Em todos os experimentos descritos a seguir, os treinamentos foram executados na *CPU* do computador. Não foram realizados testes utilizando *GPU* ou *multi-clusters*.

4.4. Experimentos

Foram realizados três experimentos com as plataformas consideradas. O primeiro considerou a rede recomendada para cada plataforma (Exp. 1). As redes recomendadas são aqueles que a documentação da plataforma sugere como opção padrão. Neste caso, apenas o conjunto de dados *MMIST* foi utilizado pois todas as plataformas o utilizam de

[4]http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Plataforma	Acurácia		Treinamento		Predição		Memória		Linhas
	Média	STD	Média	STD	Média	STD	Média	STD	
Graphlab	**0,987**	0,0008	469,7	7,52	5,54	1,57	**252,34**	2,6	**51**
H2O	0,901	0,0137	**11,3**	0,68	**0,42**	0,09	948,72	45,4	55
Singa	0,974	0,0026	295,8	21,16	1,62	0,04	643,29	0,14	110

Tabela 3. Resultados do experimento com a rede recomendada (MNIST).

Plataforma	Acurácia		Treinamento		Predição		Memória		Linhas
	Média	STD	Média	STD	Média	STD	Média	STD	
Graphlab MNIST	0,830	0,007	**28,51**	0,34	1,31	0,07	1.682,751	1,7	**45**
H2O MNIST	**0,968**	0,002	201,1	24,24	**0,69**	0,07	**896,469**	65,9	52
Graphlab KDD	0,860	0,045	**86,61**	4,783	**0,79**	0,07	**1.050,2**	9,9	**45**
H2O KDD	**0,925**	0,002	1.652,6	630,2	4,33	0,5	1.518,8	84,6	52

Tabela 4. Experimentos com redes geradas automaticamente.

forma padrão. Para obter-se valores mais fidedignos para a comparação entre as plataformas, cada teste foi executado 10 vezes. A Tabela 3 apresenta as médias de cada métrica após as 10 execuções. Nela também está incluída a coluna com a quantidade de linhas de código que foi necessária para escrever e executar os teste. Os melhores resultados (baseados nos testes estatísticos) estão destacados em negrito. Assim, observa-se que a plataforma *Singa* não teve destaque em nenhuma das métricas avaliadas. Já a plataforma *H2O* teve destaque com os menores tempos para treinamento e predição. Nas avaliações de acurácia, utilização de memória e quantidade de linhas, a plataforma *Graphlab* se mostrou melhor.

O segundo experimento focou na construção da rede neural de forma automática pela plataforma (Exp. 2). Neste caso, apenas *Graphlab* e *H2O* foram consideradas por possuirem este recurso: fazer a leitura de um conjunto de treinamento, extrair informações e a partir disso, sugerir uma rede neural de *Deep learning* automaticamente para o problema. Foram utilizados os dois conjuntos de treinamento.

Novamente, neste experimento, os modelos foram executados 10 vezes e a Tabela 4 apresenta as médias das execuções para cada métrica, além da coluna com a quantidade de linhas utilizadas para implementação. Foram destacados em negrito os melhores resultados para cada métrica de avaliação (confirmados por testes estatísticos). Assim, pode-se observar que a plataforma *H2O* tem a melhor acurácia em ambos os conjuntos, já *Graphlab* leva vantagem no tempo de treinamento e número de linhas. Na predição, cada plataforma vence uma vez.

O terceiro experimento considerou as plataformas com o uso de uma rede MLP configurada manualmente (Exp. 3). Como uma rede MLP é formada por camadas totalmente conectadas (*fully-connected*), é preciso configurar quantas serão as camadas da rede neural que compõe o modelo de *Deep learning* a ser utilizado. Sendo assim, as três plataformas foram configuradas para conter três camadas. A primeira camada, de entrada,

Plataforma	Acurácia		Treinamento		Predição		Memória		Linhas
	Média	STD	Média	STD	Média	STD	Média	STD	
Graphlab MNIST	0,940	0,002	**60,9**	1,2	1,79	0,09	1.887,9	61,2	**51**
H2O MNIST	**0,954**	0,003	138,4	2,9	0,67	0,01	**681,34**	80,9	55
Singa MNIST	0,921	0,001	139,3	6,3	**0,07**	0,03	2.815,2	0,15	110
Graphlab KDD	0,919	0,001	476,69	24,9	2,44	0,2	**1.037,4**	16,2	**51**
H2O KDD	**0,924**	0,002	**83,47**	21,1	**1,19**	0,1	1.415,6	93,1	55
Singa KDD	0,918	0,001	332,96	27,4	1,95	0,03	3.062,9	1,38	110

Tabela 5. Experimento com rede MLP configurada manualmente.

com uma unidade para cada atributo dos conjuntos de dados. Ou seja, para os testes com o conjunto MNIST, a camada de entrada ficou com 784 (28x28) unidades, e para os testes com o conjunto KDD a primeira camada ficou com 41 unidades. A segunda camada (camada oculta), foi configurada com 100 unidades, e com a saída para uma função de ativação *Sigmoid*. Por fim, a quantidade de unidades da terceira camada (*output layer*) também difere entre MNIST e KDD. Para os testes com o conjunto MNIST, a camada de saída ficou com 10 unidades, que representam os 10 possíveis rótulos (*i.e.*, dígitos de 0 a 9). Já para os testes com o conjunto KDD, a última camada ficou com 2 unidades, que representas os possíveis rótulos (*i.e.*, conexão boa ou ruim).

Os experimentos foram executados no mesmo formato dos anteriores e a Tabela 5 apresenta os resultados com as médias das execuções. Observando a tabela, percebe-se que plataforma *H2O* teve destaque na acurácia em ambos os conjuntos e a *Graphlab* para o número de linhas. No geral, a *H2O* teve cinco melhores desempenhos nos dois conjuntos de treinamento. Esses resultados vêm ao encontro dos resultados encontrados por [Ng et al. 2016], ou seja, *H20* tem um melhor desempenho sobre *Singa*. Os mesmos autores afirmam que *Singa* tem um potencial maior, mas é necessário configurar muitos hiperparâmetros para se alcançar tais desempenhos.

A Tabela 6 apresenta a classificação, por experimento, para cada uma das plataformas consideradas. Nesta tabela pode ser observado com detalhes onde cada plataforma se destacou. Percebe-se que a plataforma *Singa* foi a que apresentou a maioria dos piores desempenhos. Já as plataformas *H2O* e *Graphlab* tiveram desempenhos similares.

5. Conclusão

Depois de realizados os experimentos, foi possível perceber que a plataforma *Graphlab* é a mais simples para ser utilizada por um usuário sem muito conhecimento sobre parametrização e configuração de redes de *Deep learning*. Em oposição a isso, a plataforma *Singa* traz uma proposta para usuários mais experientes, trazendo uma maior flexibilidade de utilização e exigindo um conhecimento maior para ser devidamente configurada. De forma geral, os melhores resultados de acurácia foram gerados pela plataforma *H2O*. Principalmente quando a plataforma pôde gerar automaticamente o modelo baseando-se no conjunto de dados de treinamento, tanto para a base MNIST como para a base KDD. Nos testes que levaram em consideração o tempo de treinamento, em sua mai-

Experimento	Base	Métrica	1º colocado	2º colocado	3º colocado
Exp. 1	MNIST	Acurácia	Graphlab	Singa	H2O
		Treinamento	H2O	Singa	Graphlab
		Predição	H2O	Singa	Graphlab
		Memória	Graphlab	Singa	H2O
		Linhas	Graphlab	H2O	Singa
Exp. 2	MNIST	Acurácia	H2O	Graphlab	Singa
		Treinamento	Graphlab	H2O	Singa
		Predição	H2O	Graphlab	Singa
		Memória	H2O	Graphlab	Singa
		Linhas	Graphlab	H2O	Singa
	KDD	Acurácia	H2O	Graphlab	Singa
		Treinamento	Graphlab	H2O	Singa
		Predição	Graphlab	H2O	Singa
		Memória	Graphlab	H2O	Singa
		Linhas	Graphlab	H2O	Singa
Exp. 3	MNIST	Acurácia	H2O	Graphlab	Singa
		Treinamento	Graphlab	H2O	Singa
		Predição	Singa	H2O	Graphlab
		Memória	H2O	Graphlab	Singa
		Linhas	Graphlab	H2O	Singa
	KDD	Acurácia	H2O	Graphlab	Singa
		Treinamento	H2O	Singa	Graphlab
		Predição	H2O	Singa	Graphlab
		Memória	Graphlab	H2O	Singa
		Linhas	Graphlab	H2O	Singa

Tabela 6. *Ranking* das plataformas por métrica

oria, o destaque foi para a plataforma *Graphlab*. Já no tempo de predição, a plataforma *H2O* desempenhou a execução preditiva do conjunto de testes com o menor tempo na maioria dos experimentos. Quanto ao consumo de memória, ao utilizar-se a base MNIST, composta por imagens, o melhor desempenho foi da plataforma *H2O*. Nos testes com a base KDD que é composta por atributos alfanuméricos, a plataforma *Graphlab* conseguiu realizar o processo de treinamento consumindo a menor quantidade de memória.

Como trabalhos futuros, pretende-se ampliar o número de conjunto de dados para obter-se conclusões mais abrangentes das plataformas. Como exemplo de diversidade dos dados pode-se citar: fator de balanceamento das classes, complexidade dos atributos, dimensões dos conjunto de dados, presença ou ausência de *outliers* e número de valores faltantes. Também, pretende-se experimentar as plataformas de forma distribuída utilizando *clusters* separados fisicamente.

Referências

Bahrampour, S., Ramakrishnan, N., Schott, L., and Shah, M. (2015). Comparative study of deep learning software frameworks. In *arXiv preprint arXiv:1511.06435*.

Candel, A., Parmar, V., LeDell, E., and Arora, A. (2016). Deep learning with H2O. *H2O.ai, Inc*.

Deng, L. and Yu, D. (2014). Deep learning: methods and applications. *Foundations and Trends in Signal Processing*, 7(3–4):197–387.

Duarte, D. and Ståhl, N. (2019). Machine learning: a concise overview. In *Data Science in Practice*, pages 27–58. Springer.

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., and Lew, M. S. (2016). Deep learning for visual understanding: A review. *Neurocomputing*, 187:27 – 48.

Kovalev, V., Kalinovsky, A., and Kovalev, S. (2016). Deep learning with Theano, Torch, Caffe, TensorFlow, and Deeplearning4J: Which one is the best in speed and accuracy? In *XIII International Conference on Pattern Recognition and Information Processing*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Low, Y., Bickson, D., Gonzalez, J., Guestrin, C., Kyrola, A., and Hellerstein, J. M. (2012). Distributed GraphLab: a framework for machine learning and data mining in the cloud. *Proceedings of the VLDB Endowment*, 5(8):716–727.

Ng, S. S. Y., Zhu, W., Tang, W. W. S., Wan, L. C. H., and Wat, A. Y. W. (2016). An independent study of two deep learning platforms - H2O and Singa. In *2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*.

Ooi, B. C., Tan, K.-L., Wang, S., Wang, W., Cai, Q., Chen, G., Gao, J., Luo, Z., Tung, A. K. H., Wang, Y., Xie, Z., Zhang, M., and Zheng, K. (2015). SINGA: A distributed deep learning platform. In *ACM Multimedia*.

Shatnawi, A., Al-Bdour, G., Al-Qurran, R., and Al-Ayyoub, M. (2018). A comparative study of open source deep learning frameworks. In *9th ICICS*, pages 72–77. IEEE.

Zimmerman, D. W. (1997). Teacher's corner: A note on interpretation of the paired-samples t test. *Journal of Educational and Behavioral Statistics*, 22(3):349–360.