SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
VOL. 3, NO. 3, DECEMBER 2020
DOI: 10.35377/saucis.03.03.776573

A Comparison of the State-of-the-Art Deep Learning Platforms: An Experimental Study

Abdullah Talha Kabakus[1]
[1]Corresponding Author; Duzce University; talhakabakus@duzce.edu.tr; +90 380 542 10 36

Received 03 August 2020; Revised 15 September 2020; Accepted 18 September 2020; Published online 30 December 2020

Abstract

Deep learning, a subfield of machine learning, has proved its efficacy on a wide range of applications including but not limited to computer vision, text analysis and natural language processing, algorithm enhancement, computational biology, physical sciences, and medical diagnostics by producing results superior to the state-of-the-art approaches. When it comes to the implementation of deep neural networks, there exist various state-of-the-art platforms. Starting from this point of view, a qualitative and quantitative comparison of the state-of-the-art deep learning platforms is proposed in this study in order to shed light on which platform should be utilized for the implementations of deep neural networks. Two state-of-the-art deep learning platforms, namely, (i) *Keras*, and (ii) *PyTorch* were included in the comparison within this study. The deep learning platforms were quantitatively examined through the models based on three most popular deep neural networks, namely, (i) Feedforward Neural Network (FNN), (ii) Convolutional Neural Network (CNN), and (iii) Recurrent Neural Network (RNN). The models were evaluated on three evaluation metrics, namely, (i) *training time*, (ii) *testing time*, and (iii) *prediction accuracy*. According to the experimental results, while *Keras* provided the best performance for both FNNs and CNNs, *PyTorch* provided the best performance for RNNs expect for one evaluation metric, which was the *testing time*. This experimental study should help deep learning engineers and researchers to choose the most suitable platform for the implementations of their deep neural networks.

Keywords: deep learning, deep neural networks, feedforward neural networks, convolutional neural networks, recurrent neural networks

En Gelişkin Derin Öğrenme Platformlarının Bir Karşılaştırması: Deneysel Bir Çalışma

Öz

Makine öğrenmesinin bir alt alanı olan derin öğrenme, bilgisayarlı görü, metin analizi ve doğal dil işleme, algoritma iyileştirme, hesaplamalı biyoloji, fen bilimleri ve hastalık teşhisi alanlarıyla sınırlı olmamak kaydıyla çok çeşitli uygulamalar üzerindeki etkinliğini en gelişkin yaklaşımlardan daha başarılı sonuçlar üreterek kanıtlamıştır. Derin sinir ağlarının gerçekleştiriminde çeşitli en gelişkin platformlar mevcuttur. Bu noktadan hareketle, derin sinir ağların gerçekleştiriminde hangi platformun kullanılması gerektiğine ışık tutmak amacıyla en gelişkin derin öğrenme platformlarının nitel ve nicel bir karşılaştırması bu çalışmada öne sürülmüştür. Bu çalışma kapsamındaki karşılaştırmaya iki en gelişkin derin öğrenme platformu, isim olarak, (i) *Keras* ve (ii) *PyTorch* dahil edilmiştir. Derin öğrenme platformları en popüler üç derin sinir ağı olan (i) İleri Beslemeli Sinir Ağı (FNN), (ii) Evrişimli Sinir Ağı (CNN) ve (iii) Tekrarlayan Sinir Ağı (RNN) temelli modeller üzerinden incelenmiştir. Modeller, (i) *eğitim süresi*, (ii) *test süresi* ve (iii) *tahmin doğruluğu* olmak üzere üç değerlendirme kriteri kullanılarak değerlendirilmiştir. Elde edilen deneysel sonuçlara göre hem FNN hem de CNN'ler için en iyi performansı *Keras* sağlarken, RNN'ler için bir değerlendirme kriteri (*test süresi*) dışında en iyi performansı *PyTorch* sağlamıştır. Bu deneysel çalışma, derin öğrenme mühendisleri ve araştırmacılarının kendi derin öğrenme ağlarının gerçekleştiriminde en uygun platformun seçimi noktasında yardım etmesi gerekmektedir.

Anahtar Kelimeler: derin öğrenme, derin sinir ağları, ileri beslemeli sinir ağları, evrişimli sinir ağları, tekrarlayan sinir ağları

1. Introduction

Deep learning, a subfield of machine learning, is the application of multi-layered neural networks to perform learning tasks such as classification, regression, clustering, and auto-encoding. Deep learning has been a revolution for various learning tasks including but not limited to computer vision [1], medical diagnostics [2], text analysis and natural language processing (NLP) [3], algorithm enhancement, computational biology, and physical sciences [4] due to its efficacy in approximating and reducing huge datasets into highly accurate predictive and transformational output [5], [6]. Deep learning has even exceeded human abilities in areas such as handwriting and image recognition [7], [8]. Unlike the traditional machine learning techniques, deep learning architectures are flexible enough to be applied to different types of data, be they visual, audio, numerical, text, or some combination of them [4]. Despite that the fundamentals of the deep learning techniques were originally proposed in the 1980s, the rise in popularity of it can be traced back to only the last few years due to the following reasons: (i) The greater availability of big data, which has significantly improved learning ability of deep neural networks, thanks to the rise of smartphones, social media applications, and embedded sensors, (ii) the efficient use of graphical processing units (GPUs), and (iii) the discovery of the new architectures as well as new techniques to improve the performance of models such as *ReLU*, *Batch Normalization*, and *Dropout* [4], [9]–[14]. When it comes to implementation of deep neural networks, there exist various highly-popular, state-of-the-art platforms, which do have similar qualitative abilities, such as *Keras* [15], *PyTorch* [16], *Caffe* [17], *Theano* [18], and the *Microsoft Cognitive Toolkit (CNTK)* [19]. Therefore, which one should be utilized to implement a deep neural network is a question that instinctively comes to mind for the researchers, and developers and is needed to be addressed. To this end, a comparison, that both quantitatively and qualitatively compare the state-of-the-art deep learning platforms, was proposed in this study. This experimental study should help deep learning engineers and researchers to choose the most suitable platform for the implementations of their deep neural networks. The rest of the paper is structured as follows: Section 2 describes the related work. Section 3 presents the material and method. Section 4 presents the experimental results and discussion. Finally, Section 5 concludes the paper with future directions.

2. Related Work

Liu *et al.* [20] benchmarked three state-of-the-art deep learning platforms, namely, *TensorFlow* [21], *Caffe*, and *Torch* [22]. The evaluation metrics they used were accuracy, runtime performance, and the model's robustness against different datasets. They highlighted three observations from their experiments: (i) The deep learning platforms are optimized for the built-in datasets with their default configuration. Hence, the efficacy might vary on a custom dataset. (ii) The efficacy might vary on the dataset that was used for the experiments. (iii) Benchmarking deep learning platforms is significantly more challenging than traditional performance-driven benchmarking.

Bahrampour *et al.* [23] proposed a comparative study of *Caffe*, *neon* [24], *Theano*, and *Torch* for deep learning tasks. The three aspects they utilized were: (i) *extensibility*, (ii) *hardware utilization*, and (iii) *speed*, which includes both gradient computation time (*a.k.a.* training time) and forward time (*a.k.a.* testing time). According to their experimental result, *Torch* provided the best performance for any deep neural network architecture on CPU. When it comes to performance on GPU, the conclusions were two-fold: (i) *Torch* provided the best performance for large convolutional and fully connected networks, and (ii) *Theano* provided the best performance for LSTM (Long Short-Term Memory) networks.

Shi *et al.* [25] benchmarked four state-of-the-art deep learning platforms, namely, *Caffe*, *CNTK*, *TensorFlow*, and *Torch* for three types of neural networks, namely, (i) Feedforward Neural Network (FNN), (ii) Convolutional Neural Network (CNN), and (iii) Recurrent Neural Network (RNN). They evaluated the aforementioned deep learning platforms based on their running time performance. According to their experiments, they concluded that there is no single platform that consistently outperforms others. For the FNNs, *Torch* provided the best performance on CPU. When it comes to the performances of FNNs on GPU, *Caffe*, and *CNTK* provided the best performance. For the CNNs, while *Caffe* provided the best performance on a quad-core desktop CPU with 4 threads, *TensorFlow* provided

the best performance on a server CPU with 16 threads. When it comes to the performances of CNNs on GPU, the best performance varies through the CNN model. For the RNNs, *CNTK* provided the best performance both on CPU and GPU. Also, they noted that the performances of the deep neural networks generally do not scale very well on many-core CPUs and $10-30$X speedup was observed when the best GPU result was compared to the best CPU result.

Chintala [26], an Artificial Intelligence (AI) research engineer at *Facebook*, proposed an extensive set of benchmarks for a variety of CNN models and benchmarked *Torch*, *TensorFlow*, and *Caffe*. The experiments were carried on a machine with the following hardware configuration: 6-core *Intel Core i7-5930K* @ 3.50GHz CPU, and *NVIDIA Titan X* GPU. According to the experimental result, *Torch* provided the best performance among the others for the *AlexNet* [7] CNN model.

Theano development team [18] benchmarked the *Theano* with *TensorFlow*, and *Torch* on three LSTM models as follows: (i) The small model consists of a single 200-unit hidden layer with a sequence length of 20, (ii) the medium model consists of a single 600-unit hidden layer with a sequence length of 40, and (iii) the large one consists of two 650-unit hidden layers with a sequence length of 50. The experiments were carried on a machine with the following hardware configuration: 6-core *Intel Core i7-5930K* @ 3.50GHz CPU, and *NVIDIA Digits DevBox* with 4 *Titan X* GPUs. All models were evaluated on the *Penn Treebank* dataset [27]. The evaluation metric was the processing speed, which includes both the forward and backward passes. According to the experimental result, while *TensorFlow* provided the best performance for the small LSTM model, *Theano* provided the best performance for both the medium and large LSTM models. *Torch* provided the worst performance for all models.

Shatnawi *et al.* [28] benchmarked *CNTK*, *TensorFlow*, and *Theano* using CNNs on two gold standard datasets, namely, *MNIST (Mixed National Institute of Standards and Technology)* [29], and *CIFAR-10* [30]. According to the experimental result, *CNTK* provided the best performance among the others in terms of CPU and GPU multithreading, but in *CIFAR-10* using 8, 16, and 32 threads in CPU, *TensorFlow* was found as faster than *CNTK*. *Theano* was found as the slowest among the others.

Kovalev *et al.* [31] benchmarked *Theano* (with *Keras* wrapper), *TensorFlow*, *Caffe*, *Torch*, and *Deeplearning4j* [32] for FNNs. The evaluation metrics were processing speed, classification accuracy, and the number of lines of source code. According to the experimental result, the aforementioned deep learning platforms were ranked as follows: *Theano*, *TensorFlow*, *Caffe*, *Torch*, and *Deeplearning4j*. In addition to this, they reported that the employment of the non-linear activation function *Rectified Linear Unit (ReLU)* instead of the *tanh* activation function improved the performances of FNNs in terms of both training speed and classification accuracy.

3. Material and Method

In this section, the deep learning platforms and the benchmarking setup were described in the following subsections.

3.1 Deep Learning Platforms

The properties of deep learning platforms such as the programming languages they are implemented in, supported programming languages, *NVIDIA CUDA Deep Neural Network (cuDNN)* [33] support, which is a GPU-accelerated library of primitives for deep neural networks that provides significant speed and space benefits [34], and CPU and GPU support vary through the platforms. Table 1 lists the properties of the widely-used, state-of-the-art deep learning platforms, namely, *Keras*, *PyTorch*, *Caffe*, *Theano*, and *CNTK*. Each deep learning platform is briefly described in the following paragraphs.

Keras. *Keras* is a widely-used, open-source deep learning library implemented in Python. *Keras* provides an easy-to-use, developer-friendly API to implement deep neural network architectures. *Keras* was originally developed by a *Google* engineer and aims easy and fast prototyping [15]. Unlike the other aforementioned platforms, *Keras* is not a standalone deep learning platform as it runs on the top of

various backends, namely, *TensorFlow*, *Theano*, and *CNTK*. *TensorFlow* was employed as the backend of *Keras* within this study since it is the recommended one by its developer [35].

PyTorch. *PyTorch* is another widely-used, open-source deep learning library implemented in Python. *PyTorch* is backed by *Facebook AI Research* and behaves like a Python API for the *Torch* engine, which is written in Lua programming language and initially only had bindings in Lua [36]. While *PyTorch* retains the flexibility of interfacing with C and the current speed of the *Torch* engine, it has some big advantages such as recurrent nets, weight sharing, and memory usage [37]. Another advantage of *PyTorch* compared to *Torch* comes from being a Python library as 78% of over 23,000 data scientists recommended Python for an aspiring data scientist to learn in a recent survey [38]. As a natural consequence of this, all the deep learning platforms, that are included in this study, provide a Python API. Moreover, some of them, namely, *Keras*, *PyTorch*, and *Theano*, are actually implemented in Python.

Caffe. *Caffe* is an open-source deep learning library implemented in Python. *Caffe* is developed by the *Berkeley Vision and Learning Center (BVLC)* and is implemented in C++. It is reported that *Caffe* is able to process 40 million images per day which equals almost 2.5 ms per image when it is accelerated by a single *NVIDIA K40* or *Titan* GPU [17]. It is worth to mention that the next version of *Caffe*, *Caffe2,* has become a part of *PyTorch* in 2018 [39].

Theano. *Theano* is an open-source deep learning library implemented in Python and developed by *Mila Research Institute* as a compiler for mathematical expressions that optimize and evaluate the expressions in the syntax of *NumPy* [40], which is a widely-used Python library that provides multi-dimensional arrays and matrices, and a large collection of high-level mathematical functions to operate on these data structures. *Theano* is in a maintenance mode as its developers declared that they stopped the development of new features [41].

CNTK. *CNTK* is an open-source deep learning library implemented in C++ and developed by *Microsoft Research*. The developers of *CNTK* report that *CNTK* efficiently removes the duplicated computations in forward and backward passes, uses minimal memory, and reduces memory reallocation by reusing them [42]. *CNTK* provides APIs in both Python and C# programming languages. It is worth to mention that, similar to *Theano*, there are no plans for new feature development for *CNTK* since its latest stable release, 2.7, which was released in April 2019 [43].

Table 1 The properties of the widely-used, state-of-the-art deep learning platforms

Property	***Keras***	***PyTorch***	***Caffe***	***Theano***	***CNTK***
Core	Python	Python	C++	Python	C++
Multi-core CPU support	Available	Available	Available	Available	Available
Many-core GPU support	Available	Available	Available	Available	Available
NVIDIA cuDNN support	Available	Available	Available	Available	Available
Supported programming languages	Python	Python, C++, Java	Python	Python	Python, C#
Number of stars received on *GitHub*	$48.9k$	$40.2k$	$30.6k$	$9.2k$	$16.8k$

The popularities of the aforementioned deep learning platforms were retrieved through *Google Trends* [44], which is a service by *Google* that analyzes the popularities of the given terms. As the worldwide trends of the deep learning platforms in the last 5 years were presented in Figure 1, the rank of the popularities of the deep learning platforms was found as follows: *Keras*, *PyTorch*, *Caffe*, *Theano*, and *CNTK*, whose average trend scores were obtained as 56, 33, 15, 5, and 2, respectively. For the sake of comparison, the two most popular deep learning frameworks in terms of (i) the number of stars received on *GitHub*, and (ii) the trend scores which were obtained from *Google Trends*, namely, *Keras*, and *PyTorch*, were benchmarked within this study. The benchmarking experiments within this study were

carried out on the *Google*'s *Colaboratory* (*a.k.a. Colab*) [45] platform, which provides free powerful GPUs such as *Nvidia Tesla K80* as high computational power is necessary to train deep neural networks with a large amount of data. Another advantage of utilizing the *Colab* is that many highly popular Python libraries including but not limited to *TensorFlow*, *Keras*, *PyTorch*, *NumPy*, *Pandas*, and *scikit-learn* are already pre-installed on this platform. The versions of *Keras* and *PyTorch* were 2.3.1 on the *TensorFlow* 2.2.0 backend, and 1.6.0, respectively. The operating system of the host provided by *Colab* was GNU/Linux 4.19.104 *x86_64* which was bundled with Python 3.6.9.

Figure 1 The trend scores of the deep learning platforms which were obtained from *Google Trends* in the last **5** years

3.2 Benchmarking Setup

For the sake of benchmarking the deep learning platforms, models based the three most popular types of deep neural networks, namely, FNN, CNN, and RNN, were proposed and trained on the de-facto standard datasets since datasets play a critical role in the performance of deep neural networks [5], [46]–[48].

Feedforward Neural Networks. In order to benchmark the performance of *Keras* and *PyTorch* on FNNs, a sample model, whose architecture's block representation is presented in Figure 2, was implemented using these deep learning platforms.

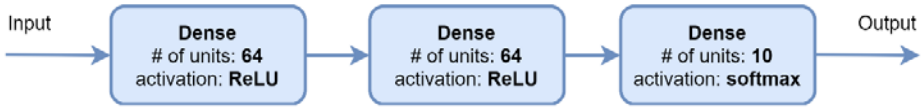

Figure 2 A block representation of the architecture of the proposed sample FNN model

In order to train and test the network, a de-facto standard dataset, namely, *MNIST*, was utilized. *MNIST* is a large dataset of handwritten digits that were size-normalized and centered in a fixed-size as some examples of the images in the dataset are presented in Figure 3. Each digit in *MNIST* is represented as a $28x28$ pixel grayscale image. This dataset is already provided by both *Keras* and *PyTorch* through the $keras$, and $torchvision$ packages, respectively. To prevent any potential issues due to manual installation, the built-in versions of the *MNIST* were preferred. The *Adaptive Moment Estimation (Adam)* [49], which is an extension to the *Stochastic Gradient Descent (SGD)* [50], was employed as the optimization algorithm of the proposed sample FNN model with the intention of updating the network weights more efficiently by computing adaptive learning rates for each network parameter from estimates of first and second moments of the gradient [2]. The hyper-parameters of the proposed sample FNN model are listed in Table 2.

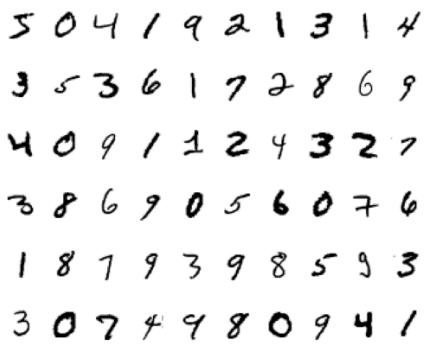

Figure 3 Some examples of the images in the *MNIST* dataset

Table 2 The hyper-parameters of the proposed sample FNN model

Hyper-parameter	**Value**
Optimization algorithm	*Adam*
Learning rate	e^{-3}
Loss function	*Categorical Cross-Entropy*
Batch size	80
Number of epochs	20

Convolutional Neural Networks. In order to benchmark the performance of *Keras* and *PyTorch* on CNNs, a highly popular architecture, namely, *VGG16* [51], was utilized which achieved 92.7% top-5 accuracy for the gold standard *ImageNet* [1] dataset. Both *Keras* and *PyTorch* provide *VGG16* implementations through the *keras*, and *torchvision* packages, respectively. A block representation of the architecture of the *VGG16* is presented in Figure 4.

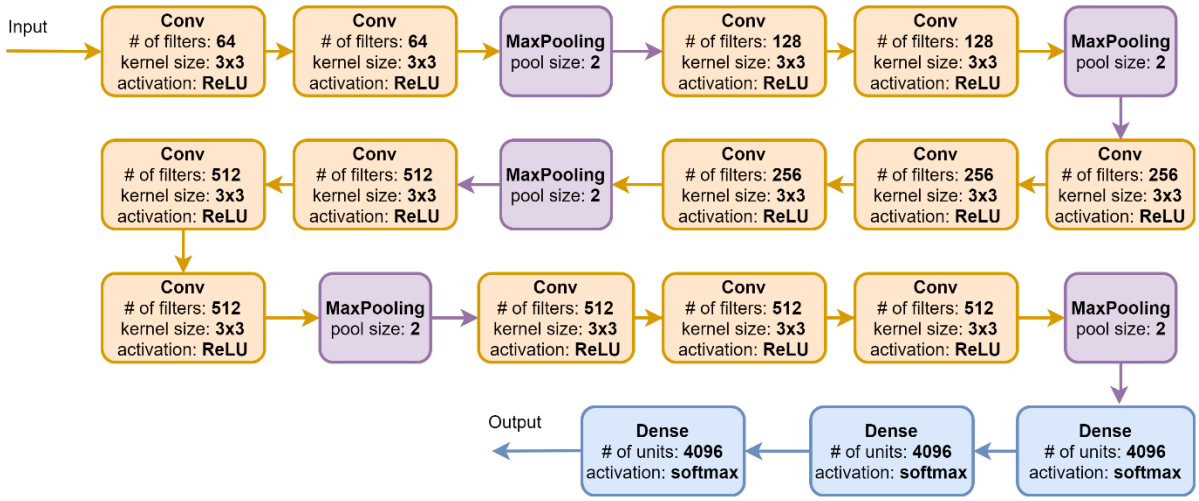

Figure 4 A block representation of the architecture of the *VGG16*

In order to train and test the network, a de-facto standard dataset, namely, *CIFAR-10*, was utilized. *CIFAR-10* is a large database of color images in ten classes, namely, *airplane*, *automobile*, *bird*, *cat*, *deer*, *dog*, *frog*, *horse*, *ship*, and *truck*. Each sample is represented as a $32x32$ pixel color image as some examples of the images in the dataset are presented in Figure 5. This dataset is already provided by both *Keras* and *PyTorch* through the *keras*, and *torchvision* packages, respectively. Similar to the experiment on FNNs, the built-in versions of the *CIFAR-10* were preferred in order to prevent any potential issues due to manual installation. The employed hyper-parameters of the *VGG16* are listed in Table 3.

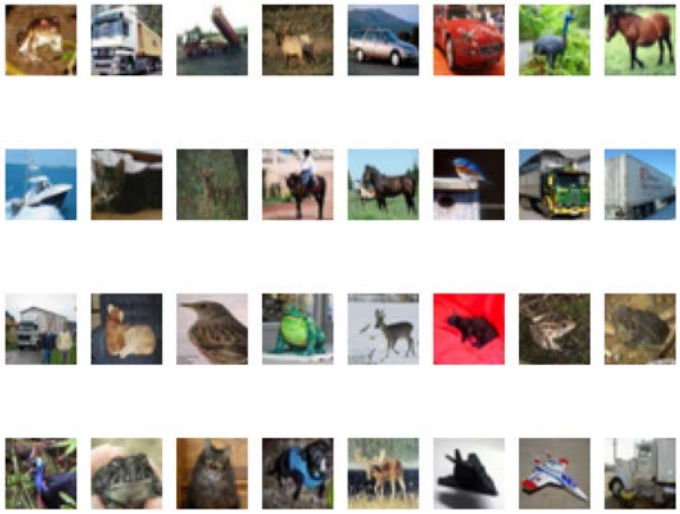

Figure 5 Some examples of the images in the *CIFAR-10* dataset

Table 3 The employed hyper-parameters of the *VGG16*

Hyper-parameter	**Value**
Optimization algorithm	*Adam*
Learning rate	e^{-2}
Loss function	*Categorical Cross-Entropy*
Batch size	80
Number of epochs	20

Recurrent Neural Networks. LSTM is a special type of RNN that provides the following advantages comparing to RNNs: (i) LSTM solves the general problem of gradient descent [52], and (ii) it has long-term memory, which is a key necessity for sequence processing. In order to benchmark the performance of *Keras* and *PyTorch* on RNNs, a sample LSTM model, whose architecture's block representation is presented in Figure 6, was implemented using these deep learning platforms.

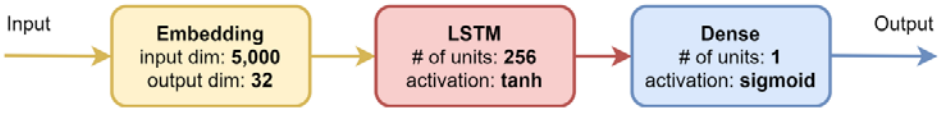

Figure 6 A block representation of the architecture of the proposed sample LSTM model

In order to train and test the network, a de-facto standard dataset, namely, *IMDb Movie Review* [53] dataset, was utilized. This dataset consists of movie reviews from *IMDb* (*Internet Movie Database*), a widely-used online movie database. Each movie review in the dataset is encoded as a list of word indexes ($integers$) and is labeled with a sentiment class ($positive$/$negative$). Some samples from the *IMDb Movie Review* dataset are listed in Table 4.

Table 4 Some samples from the *IMDb Movie Review* dataset

Movie Review	**Sentiment Class**
"If you like original gut wrenching laughter you will like this movie. If you are young or old then you will love this movie, hell even my mom liked it. Great Camp!!!"	$positive$
"This movie was terrible. The plot was terrible and unbelievable. I cannot recommend this movie. Where did this movie come from? This movie was not funny and wasted the talent of some great actors and actresses including: Gary Sinise, Kathy Bates, Joey Lauren Adams, and Jennifer Tilly."	$negative$

The *IMDb Movie Review* dataset is already provided by both *Keras* and *PyTorch* through the $keras$, and $torchtext$ packages, respectively. Similar to the previous experiments, the built-in versions of the

IMDb Movie Review dataset were preferred in order to prevent any potential issues due to manual installation. The hyper-parameters of the proposed sample LSTM model are listed in Table 5.

Table 5 The hyper-parameters of the proposed sample LSTM model

Hyper-parameter	**Value**
Optimization algorithm	*Adam*
Learning rate	e^{-2}
Loss function	*Binary Cross-Entropy*
Batch size	500
Number of epochs	10

4. Experimental Result and Discussion

All the experiments were evaluated on the GPUs available on *Colab* since the significant processing speedup of deep neural networks as a result of the utilization of GPUs instead of CPUs is widely experimented [20], [23], [25], [28]. Evaluation metrics are critical for benchmarking studies. The following three evaluation metrics were used in this study: (i) *Training time*, the time spent on training the network, (ii) *testing time*, the time spent on testing the trained network which is a clear indicator of any potential latency of deploying the model for prediction [20], and (iii) *prediction accuracy*, the accuracy of the model for predicting the unknown samples (*a.k.a.* testing set). It is worth to mention that these durations were calculated thanks to the built-in Python function $time$, which is available in the $time$ package of the Python SDK and returns the current time in seconds since the Epoch, through the calculation of the time difference between the timestamps retrieved before and after each phase (training/testing) of the employed networks. Also, each experiment was repeated 10 times and the final values were determined through the cumulative averages of the trials. In the following paragraphs, the experimental result and discussion are presented for each neural network type.

Feedforward Neural Networks. *MNIST* dataset was utilized to train and test the proposed FNN model for the sake of benchmarking the deep learning platforms on FNNs. *MNIST* consists of $\mathbf{60,000}$ training, and $\mathbf{10,000}$ test images. $\mathbf{20}\%$ of the training images were employed as the validation set which is necessary to update the weights and tune the model. *Keras* was found as more accurate than *PyTorch* on prediction accuracy as the experimental result is listed in Table 6. When it comes to training time, *Keras* was found about $\mathbf{3.8}$ times faster than *PyTorch*. For the testing time, *Keras* was found about $\mathbf{2.4}$ times faster than *PyTorch*. The calculated training and testing times of *Keras* and *PyTorch* for the proposed sample FNN model are presented in Figure 7. According to this experiment, it is safe to conclude that *Keras* is a better choice for the implementations of FNNs.

Table 6 The calculated prediction accuracy of *Keras* and *PyTorch* for the proposed sample FNN model

Platform	**Accuracy (%)**
Keras	97.24
PyTorch	96.69

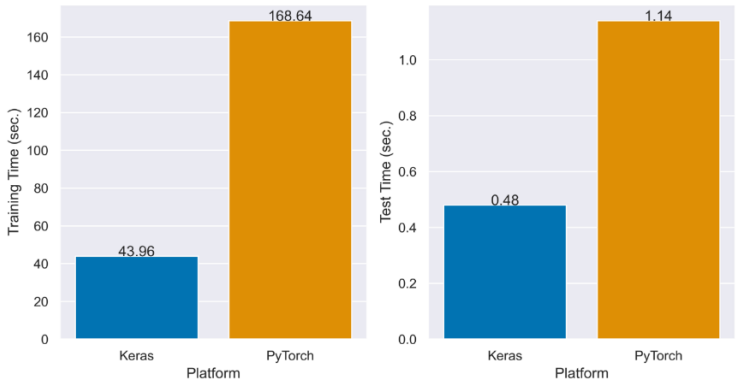

Figure 7 The calculated training (left) and testing (right) times of *Keras* and *PyTorch* for the proposed sample FNN model

Convolutional Neural Networks. The *CIFAR-10* dataset was utilized to train and test the employed *VGG16* for the sake of benchmarking the deep learning platforms on CNNs. *CIFAR-10* consists of $\mathbf{50,000}$ training, and $\mathbf{10,000}$ test images. $\mathbf{20}$% of the training images were employed as the validation set which is necessary to update the weights and tune the model during backpropagation. *Keras* was found as more accurate than *PyTorch* on prediction accuracy as the experimental result is listed in Table 7. When it comes to training time, *Keras* was found about $\mathbf{1.9}$ times faster than *PyTorch*. For the testing time, *Keras* was found about $\mathbf{1.4}$ times faster than *PyTorch*. The calculated training and testing times of *Keras* and *PyTorch* for the employed *VGG16* are presented in Figure 8. Consequently, it is safe to conclude from this experiment that *Keras* was found as a better choice for the implementations of CNNs.

Table 7 The calculated prediction accuracy of *Keras* and *PyTorch* for the employed *VGG16*

Platform	**Accuracy (%)**
Keras	78.43
PyTorch	76.54

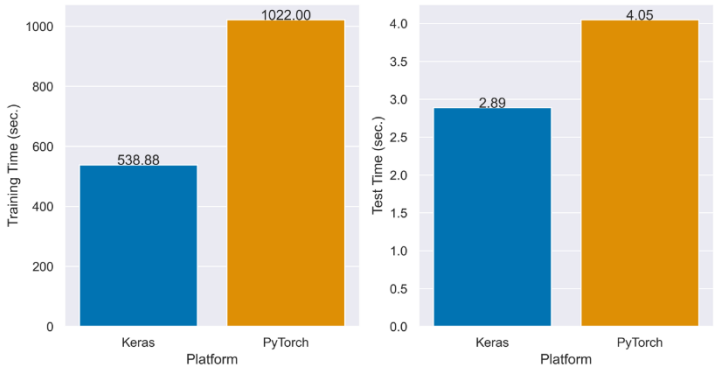

Figure 8 The calculated training (left) and testing (right) times of *Keras* and *PyTorch* for the employed *VGG16*

Recurrent Neural Networks. The *IMDb Movie Review* dataset was utilized to train and test the proposed sample LSTM model for the sake of benchmarking the deep learning platforms on RNNs. The *IMDb Movie Review* dataset consists of $\mathbf{25,000}$ movie reviews for training, and $\mathbf{25,000}$ movie reviews for testing, and only top (most frequent $\mathbf{5,000}$) words were kept. $\mathbf{20}$% of the training images, $\mathbf{5,000}$ movie reviews, were employed as the validation set. *PyTorch* was found as more accurate than *Keras* as the experimental result is listed in Table 8. When it comes to training time, *PyTorch* was found about $\mathbf{1.3}$ times faster than *Keras*. Unlike training, *Keras* was found about $\mathbf{1.6}$ times faster than *PyTorch* for testing. The calculated training and testing times of *Keras* and *PyTorch* for the proposed sample LSTM model are presented in Figure 9. According to this experiment, it is safe to conclude that *PyTorch* was found as a better choice for the implementations of RNNs as *Keras* was found better at only one of the evaluation metrics, which was the *testing time*.

Table 8 The calculated prediction accuracy of *Keras* and *PyTorch* for the proposed sample LSTM model

Platform	**Accuracy (%)**
Keras	85.83
PyTorch	87.08

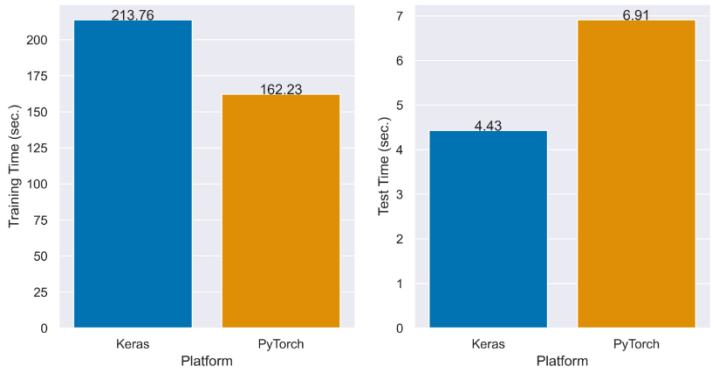

Figure 9 The calculated training (left) and testing (right) times of *Keras* and *PyTorch* for the employed *VGG16*

5. Conclusion

Deep neural networks have proven their efficacy in many topics and their effectiveness is still being experimented on a wide range of topics thanks to the previous great success. Since there exist various highly-popular, state-of-the-art platforms for the implementation of deep neural networks, which one provides the best performance is a question that should be shed light on. To this end, five state-of-the-art deep neural network platforms, namely, (i) *Keras*, (ii) *PyTorch*, (iii) *Caffe*, (iv) *Theano*, and (v) *CNTK* were compared in this study. The two most popular of these platforms, namely, *Keras*, and *PyTorch*, were both quantitatively and qualitatively compared. For the quantitative comparison, models that were based on three widely-used deep neural network types, namely, (i) FNN, (ii) CNN, and (iii) RNN, were implemented using *Keras* and *PyTorch*. Three evaluation metrics, namely, (i) *training time*, (ii) *testing time*, and (iii) *prediction accuracy*, were used for the performance comparison of the deep neural network platforms. According to the experimental result, *Keras* was found as a better choice both accuracy-wise and time-wise compared to *PyTorch* for the models based on FNNs and CNNs. When it comes to models based on RNNs, while *PyTorch* provided better accuracy and required less time to train the model, *Keras* was found as faster than *PyTorch* for the testing of RNNs.

As future work, the proposed models can be employed on CPU to reveal their performances under CPU. Also, more deep neural network types and more deep neural network platforms can be included for the conducted experiments for a more comprehensive benchmark. In addition to this, the technical reasons behind the performance differences between the deep learning platforms can be further investigated by deeply investigating the implementations of these platforms. Finally, the qualities of deep neural network platforms can be evaluated with respect to distributed-execution.

References

[1] O. Russakovsky et al., "ImageNet Large Scale Visual Recognition Challenge," *Int. J. Comput. Vis.*, vol. 115, pp. 211–252, 2015, doi: 10.1007/s11263-015-0816-y.

[2] N. Brancati, G. De Pietro, M. Frucci, and D. Riccio, "A Deep Learning Approach for Breast Invasive Ductal Carcinoma Detection and Lymphoma Multi-Classification in Histological Images," *IEEE Access*, vol. 7, pp. 44709–44720, 2019, doi: 10.1109/ACCESS.2019.2908724.

[3] Y. Weng, F. Bell, H. Zheng, and G. Tur, "OCC: A Smart Reply System for Efficient In-App Communications," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*, 2019, pp. 1–8, doi: 10.1145/3292500.3330694.

[4] W. G. Hatcher and W. Yu, "A Survey of Deep Learning: Platforms, Applications and Emerging Research Trends," *IEEE Access*, vol. 6, pp. 24411–24432, 2018, doi: 10.1109/ACCESS.2018.2830661.

[5] X. W. Chen and X. Lin, "Big Data Deep Learning: Challenges and Perspectives," *IEEE Access*, vol. 2, pp. 514–525, 2014, doi: 10.1109/ACCESS.2014.2325029.

[6] N. D. Nguyen, T. Nguyen, and S. Nahavandi, "System Design Perspective for Human-Level Agents Using Deep Reinforcement Learning: A Survey," *IEEE Access*, vol. 5, pp. 27091–27102, 2017, doi: 10.1109/ACCESS.2017.2777827.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Proceedings of the 25th International Conference on*

Neural Information Processing Systems - Volume 1 (NIPS'12), 2012, pp. 1097–1105.

[8] M. Nielsen, "Neural Networks and Deep Learning," 2019. http://neuralnetworksanddeeplearning.com (accessed Sep. 03, 2020).

[9] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[10] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML 2010)*, 2010, pp. 807–814.

[11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nat. Methods*, vol. 521, pp. 436–444, 2015, doi: 10.1038/nmeth.3707.

[12] P. Goldsborough, "A Tour of TensorFlow," *arXiv Prepr.*, vol. 1610.01178, pp. 1–16, 2016.

[13] L. Rampasek and A. Goldenberg, "TensorFlow: Biology's Gateway to Deep Learning?," *Cell Syst.*, vol. 2, no. 1, pp. 12–14, 2016, doi: 10.1016/j.cels.2016.01.009.

[14] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, 2015, pp. 448–456.

[15] F. Chollet, "Keras: the Python deep learning API," 2015. https://keras.io (accessed Sep. 03, 2020).

[16] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Proceedings of the Thirty-third Conference on Neural Information Processing Systems (NIPS 2019)*, 2019, pp. 8026–8037.

[17] Y. Jia et al., "Caffe: Convolutional Architecture for Fast Feature Embedding," in *Proceedings of the 22nd ACM International Conference on Multimedia (MM 2014)*, 2014, pp. 675–678, doi: 10.1145/2647868.2654889.

[18] R. Al-Rfou, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv Prepr.*, vol. 1605.02688, pp. 1–19, 2016.

[19] "The Microsoft Cognitive Toolkit," Microsoft, 2017. https://docs.microsoft.com/en-us/cognitive-toolkit/ (accessed Aug. 02, 2020).

[20] L. Liu, Y. Wu, W. Wei, W. Cao, S. Sahin, and Q. Zhang, "Benchmarking Deep Learning Frameworks: Design Considerations, Metrics and Beyond," in *Proceedings of the 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS 2018)*, 2018, pp. 1258–1269, doi: 10.1109/ICDCS.2018.00125.

[21] M. Abadi et al., "TensorFlow: A System for Large-Scale Machine Learning," in *Proceedings of*

the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2016), 2016, pp. 265–283.

[22] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A Matlab-like Environment for Machine Learning," in *Proceedings of the Twenty-fifth Conference on Neural Information Processing Systems (NIPS 2011)*, 2011, pp. 1–6.

[23] S. Bahrampour, N. Ramakrishnan, L. Schott, and M. Shah, "Comparative Study of Caffe, Neon, Theano, and Torch for Deep Learning," in *Proceedings of the 4th International Conference on Learning Representations (ICLR 2016)*, 2016, pp. 1–11, doi: 10.1227/01.NEU.0000297044.82035.57.

[24] "NervanaSystems/neon: Intel® NervanaTM reference deep learning framework committed to best performance on all hardware," Intel, 2015. https://github.com/NervanaSystems/neon (accessed Aug. 02, 2020).

[25] S. Shi, Q. Wang, P. Xu, and X. Chu, "Benchmarking State-of-the-Art Deep Learning Software Tools," in *Proceedings of the 2016 7th International Conference on Cloud Computing and Big Data (CCBD 2016)*, 2016, pp. 99–104, doi: 10.1109/CCBD.2016.029.

[26] S. Chintala, "Easy benchmarking of all publicly accessible implementations of convnets," 2017. https://github.com/soumith/convnet-benchmarks (accessed Aug. 02, 2020).

[27] M. Marcus, B. Santorini, and M. Marcinkiewicz, "Building a Large Annotated Corpus of English: The Penn Treebank," *Comput. Linguist.*, vol. 19, no. 2, pp. 313–330, 1993.

[28] A. Shatnawi, G. Al-Bdour, R. Al-Qurran, and M. Al-Ayyoub, "A Comparative Study of Open Source Deep Learning Frameworks," in *Proceedings of the 2018 9th International Conference on Information and Communication Systems (ICICS 2018)*, 2018, pp. 72–77, doi: 10.1109/IACS.2018.8355444.

[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998, doi: 10.1109/5.726791.

[30] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," 2009. doi: 10.1.1.222.9220.

[31] V. Kovalev, A. Kalinovsky, and S. Kovalev, "Deep Learning with Theano, Torch, Caffe, TensorFlow, and Deeplearning4J: Which One Is the Best in Speed and Accuracy?," in *Proceedings of the 13th International Conference on Pattern Recognition and Information Processing (PRIP 2016)*, 2016, pp. 99–103.

[32] "Deeplearning4j: Deep Learning for Java," Konduit, 2020. https://deeplearning4j.org (accessed Aug. 02, 2020).

[33] "NVIDIA cuDNN," NVIDIA, 2020. https://developer.nvidia.com/cudnn (accessed Aug. 02, 2020).

[34] A. Vedaldi and K. Lenc, "MatConvNet: Convolutional Neural Networks for MATLAB," in

Proceedings of the 23rd ACM International Conference on Multimedia (MM'15), 2015, pp. 689–692.

[35] F. Chollet, Deep Learning with Python. Manning Publications, 2017.

[36] N. Ketkar, Deep Learning with Python. Springer, 2017.

[37] S. Chintala, "Roadmap for torch and pytorch," 2017. https://discuss.pytorch.org/t/roadmap-for-torch-and-pytorch/38/2 (accessed Aug. 02, 2020).

[38] B. Hayes, "Programming Languages Most Used and Recommended by Data Scientists," *Business Over Broadway*, 2019. https://businessoverbroadway.com/2019/01/13/programming-languages-most-used-and-recommended-by-data-scientists/ (accessed Aug. 02, 2020).

[39] "Caffe2 and PyTorch join forces to create a Research + Production platform PyTorch 1.0," 2018. https://caffe2.ai/blog/2018/05/02/Caffe2_PyTorch_1_0.html (accessed Aug. 02, 2020).

[40] T. E. Oliphant, A Guide to NumPy. Trelgol Publishing, 2006.

[41] Y. Bengio, "MILA and the future of Theano," 2017. https://groups.google.com/forum/#!msg/theano-users/7Poq8BZutbY/rNCIfvAEAwAJ (accessed Aug. 02, 2020).

[42] D. Yu et al., "An Introduction to Computational Networks and the Computational Network Toolkit," 2015. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2014/08/CNTKBook-20160217.pdf.

[43] "CNTK v2.7 Release Notes," Microsoft Research, 2019. https://docs.microsoft.com/en-us/cognitive-toolkit/releasenotes/cntk_2_7_release_notes (accessed Aug. 02, 2020).

[44] "Google Trends," Google, 2020. https://trends.google.com/trends (accessed Aug. 02, 2020).

[45] "Colaboratory," Google, 2020. https://colab.research.google.com (accessed Sep. 03, 2020).

[46] O. Y. Al-Jarrah, P. D. Yoo, S. Muhaidat, G. K. Karagiannidis, and K. Taha, "Efficient Machine Learning for Big Data: A Review," *Big Data Res.*, vol. 2, no. 3, pp. 87–93, 2015, doi: 10.1016/j.bdr.2015.04.001.

[47] T. Condie, P. Mineiro, N. Polyzotis, and M. Weimer, "Machine learning on Big Data," in *Proceedings of the 2013 IEEE 29th International Conference on Data Engineering (ICDE 2013)*, 2013, pp. 1242–1244.

[48] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, Big, Simple Neural Nets for Handwritten Digit Recognition," *Neural Comput.*, vol. 22, no. 12, pp. 3207–3220, 2010, doi: 10.1162/NECO_a_00052.

[49] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," in *Proceeding of the 3rd International Conference on Learning Representations (ICLR 2015)*, 2015, pp. 1–15.

[50] H. Robbins and S. Monro, "A Stochastic Approximation Method," *Ann. Math. Stat.*, vol. 22, no. 3, pp. 400–407, 1951, doi: 10.1214/aoms/1177729586.

[51] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv Prepr.*, pp. 1–14, 2014, [Online]. Available: http://arxiv.org/abs/1409.1556.

[52] H. Wang, Y. Zhang, and X. Yu, "An Overview of Image Caption Generation Methods," *Comput. Intell. Neurosci.*, vol. 2020, pp. 1–13, 2020, doi: 10.1155/2020/3062706.

[53] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning Word Vectors for Sentiment Analysis," 2011.