Variance in Open Source Deep Learning Framework Performance for Convolutional Neural Networks

Applications of TensorFlow and Keras to Differing Levels of Image Classification Complexity

Connie Fan
Booth School of Business
University of Chicago
Chicago, IL United States
connie.fan@chicagobooth.edu

Tianchu Shu
Harris School of Public Policy
University of Chicago
Chicago, IL United States
tshu@uchicago.edu

ABSTRACT

Deep learning, an important branch of machine learning, has achieved transformative success in pattern recognition applications, including computer vision, speech recognition, and robotics. A set of software frameworks have been developed and are currently available. Selections of a specific framework and hardware are both important tasks, especially when computational resources are limited and consistent performance is imperative. We evaluate variance in the performance of two of the most widely used deep learning frameworks, namely Tensorflow and Keras (with TensorFlow backend), with image classification examples on three datasets of varying complexity: MNIST, fashion MNIST and breast histology images. Computation time and accuracy are reported for both GPU settings.

KEYWORDS

Deep Learning, Framework comparison, Convolutional Neural Networks, Image classification

1 INTRODUCTION

Deep learning has resulted in effective strategies for improving performance in a large number of applications, becoming one of the most used strategies by developers and researchers. In these many strategies that are deployed in products which affect users' lives, consistency in performance may be important. In our research, we aimed to explore stability in performance for our two key frameworks of interest (Tensorflow and Keras/Tensorflow). We analyzed training time and accuracy results across two convolutional neural network architectures (detailed further in section 2d), each applied to three different datasets (MNIST[1], Fashion MNIST[2], and breast histology[3]). Each combination of framework, architecture, and data (12 total combinations) was run 15 times. We then evaluated differences in performance and consistency of performance across each of our 12 use cases.

Our initial findings, further detailed under conclusions (section 6) suggest that Keras with a TensorFlow backend has less variance in performance on the training time metric than native Tensorflow. Generally, Tensorflow could be expected to be faster than Keras. There was little variance in accuracy for either framework.

In our research, the maximum training time needed for any model was roughly twelve minutes. Even within these relatively small scale implementations, were were able to observe variations in performance across frameworks. When practitioners in industry select frameworks for large-scale projects, the findings of this paper may have broader implications on their framework selection if performance consistency is critical to their work.

2 MATERIALS AND METHODS

A. Frameworks

From a list of available deep learning software frameworks, we select two of the most widely-used frameworks.

1) TensorFlow: originally named DistBelief, It is a software library for numerical computation using data flow graphs, which was developed in 2011 by the Google Brain Research Group for conducting machine learning and deep neural networks research. We implemented low-level TensorFlow

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

[2] https://github.com/zalandoresearch/fashion-mnist

[3] http://www.andrewjanowczyk.com/use-case-6-invasive-ductal-carcinoma-idc-segmentation/

for our research in which we defined the dataflow graph and then created a TensorFlow session to run parts of the graph.

2) Keras: Keras is a high-level neural network API, written in Python and supports Tensorflow, CNTK and Theano. It was developed with a focus on enabling fast experimentation. For our research, we used Keras with a Tensorflow backend.

B. Hardware

1) Graphics Processing Unit (GPU): Google Colaboratory provides free Tesla K80 GPU, with 4992 CUDA Cores, memory size of 12GB, and memory bandwidth of 240 GB/sec. Google offers 12 hours of free usage of a GPU as a backend.

C. Datasets

There are many publicly available datasets that can be used to evaluate the accuracy of a given CNN. The simplest and most common task is image classification, which involves being given an entire image, and selecting 1 of N classes that the image most likely belongs to. For our research, we aimed to select three datasets that represented a varying range of potential complexity available in image classification tasks.

1) MNIST: This dataset is our proxy for “simple” classification tasks. The MNIST dataset is a widely used dataset for digit classification, consists of 28×28 pixel grayscale images of handwritten digits. There are 10 classes (0 to 9) and 60,000 training images and 10,000 test images. MNIST is now considered a toy sample for evaluating machine learning models.

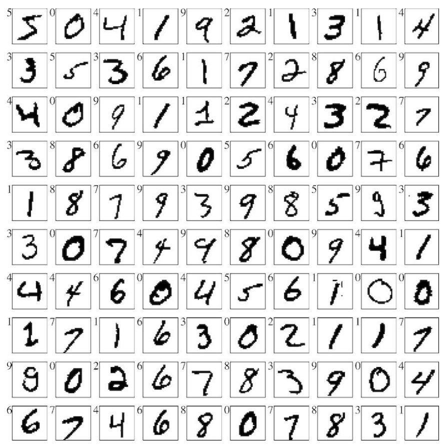

Figure 1: Image samples for MNIST (10 classes, 60k training, 10k testing)

2) Fashion MNIST: This dataset is our proxy for “moderate” classification tasks. The Fashion MNIST is a dataset of Zalando's article images, as it is shown in figure 2. It shares the same image size of MNIST (28×28). Overall, the image dataset is composed of 10 image sub-classes (i.e images taken from different objects such as T-shirt/top, trousers, pullover, dress, coat, sandal, shirt, sneaker, bag, ankle boot). The database is composed of a total of 70,000 images, from which 60,000 are conventionally used for training and 10,000 for testing.

Figure 2: Image samples for Fashion MNIST (10 classes, 60k training, 10k testing)

3) Breast Histology Images: This dataset is our proxy for “complex” classification tasks. It consists of 5547 breast histology images of size 50×50×3, curated from Andrew Janowczyk’s website. There are 2 classes, cancerous images (IDC : invasive ductal carcinoma) vs non-IDC image. Invasive Ductal Carcinoma (IDC) is the most common subtype of all breast cancers.

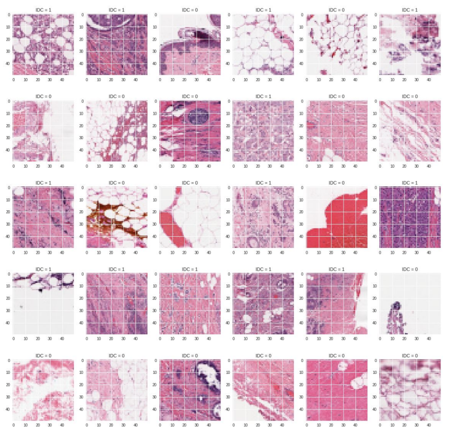

Figure 3: Image samples for Breast Histology (2 classes, 4438 training, 1110 testing)

In summary we sought to evaluate our Convolutional Neural Networks using Tensorflow and Keras with a Tensorflow backend on tasks with varying difficulty. Using the MNIST, Fashion MNIST, and Breast Histology images datasets will allow us to analyze the performance of the two frameworks at different levels of complexity. Thus in terms of evaluating the performance of a given CNN, it is important to consider that dataset upon which the performance is measured.

D. Deep Learning Architectures

With the aim of carrying out the framework performance comparison across varying architectures, two convolutional neural network architectures were implemented. The architectures were designed to represent a range of depth, types of layers (i.e., batchnorm, pool, etc.), and number of parameters. The architectures for MNIST/ Fashion MNIST and breast histology differ slightly because of different input shapes (28x28x1 vs 50x50x1) and different outputs (10 classes vs 2 classes). However, we aimed to hold as much constant as possible.

Architecture 1 performs convolutional operations by Kernels of size (5×5×16) and (5×5×32), followed by using a max pooling layer (window size 2×2) and Rectified Linear Units (ReLU) activation function for every neuron. The output was flattening out of the network to a fully connected layer with a Softmax activation function to map energy function values to discrete outputs. Architecture 1 has 28.938 trainable parameters for the MNIST/ Fashion MNIST model. Architecture 1 has 24,866 trainable parameters for the breast cancer model.

Architecture 2 goes deeper to 10 layers consisting of four CONV layers, two pooling layers, two normalization layers and one FC layer. First, it is composed of 2 hidden layers, which perform convolutional operations by Kernels of size (5×5×128) and (5×5×64), followed by using batch normalization and max pooling layers. Additionally, it performs convolutional operations by Kernels of size (5×5×32) and (5×5×16), followed by using batch normalization and max pooling layers. The output was flattening out of the network to fully connected layers, and a layer with a Softmax function to map energy function values to discrete outputs. Architecture 2 has 280,174 trainable parameters for the MNIST/ Fashion MNIST model. Architecture 2 has 284,210 trainable parameters for the breast cancer model.

Figure 4 illustrates the complete convolutional neural network implemented in this study.

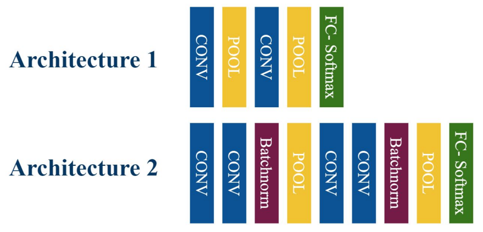

Figure 4: Network architectures for image classification task

3 PERFORMANCE METRICS

To assess performance, we evaluated two key metrics: training time and model accuracy. For both metrics, we were most interested in variance across our 15 runs of each framework/ architecture/ data combination.

A. Training Time

Training time was measured in seconds by the time() function in Python. The "timer" was started at the beginning of training and stopped after the conclusion of the final epoch.

B. Model Accuracy

To evaluate model performance, we used the misclassification rate defined by:

$$\frac{1}{n}\sum_{i=1}^{n} 1\{y_i \neq \hat{y}_i\}$$

Cross entropy loss was measured for our models using the tf.nn.softmax_cross_entropy_with_logits() function.

4 RESULTS

A. Experimental Setup

Each model was trained on 20 epochs with a batch size of 32 and learning rate of 0.01. Total training time and final model accuracy was reported for 15 runs of each framework/data/architecture combination (12 total combinations). Holding combination of framework/data/architecture constant, we reran 15 iterations on the GPU and compared results across runs. Average performance and variance across the 15 iterations is reported below.

B. Performance for Architecture 1

For Architecture 1, our simpler architecture, we observed that Keras with a TensorFlow backend took approximately 35% longer to train on the same dataset as native TensorFlow. Keras also roughly improved upon TensorFlow's classification accuracy by 2%. This result is surprising because the architecture of the model was the same across Keras and Tensorflow implementations. Because Keras was using TensorFlow as its backend, we expected the accuracy results of the Keras and Tensorflow implementations to be within a small margin of each other. Though training time differs between the two frameworks, there is negligible variance within the accuracy of each framework (Figure 10),

Below, we share a summary of our results and a statistical analysis of the observed data collected from consecutive runs of Architecture 1 on the various frameworks and datasets.

As shown in Figure 5, TensorFlow outperformed Keras in training time on the MNIST and Fashion MNIST datasets. Keras training time had greater variance for MNIST while TensorFlow training time had greater variance for Fashion MNIST. Keras provided higher accuracy across both datasets, on average.

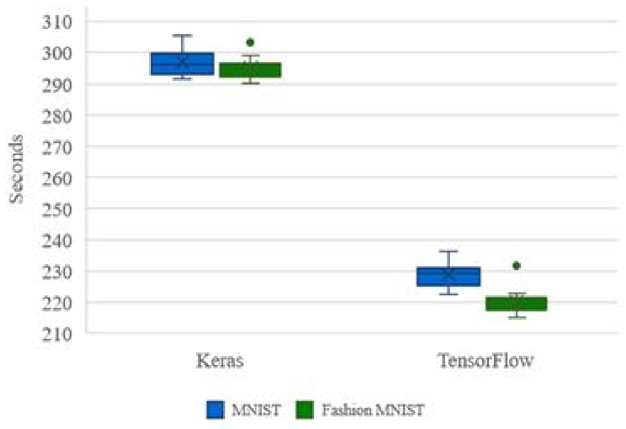

Figure 5: Architecture 1 MNIST/ Fashion MNIST Training Time

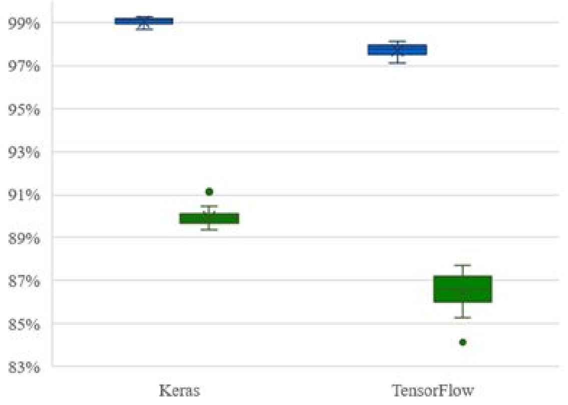

Figure 6: Architecture 1 MNIST/ Fashion MNIST Accuracy

As shown in Figure 7, TensorFlow outperformed Keras in training time on the breast cancer dataset, on average. TensorFlow training time had greater variance and also provided higher accuracy. While TensorFlow has greater variance, Keras had unexplained outliers. We suspect that the variances and outliers in our results are impacted by the way in which Colab allocates resources on the backend to our instance. Things that could affect Colab's allocations include: time of day (we ran during the work day as well as ~3am CT- training time during core North American business hours was longer than training time at 3am CT), number of parallel experiments run under one IP subnet, and efficiency of implementation of our algorithms. If given the opportunity for additional research to reproduce our experiments, given unlimited resources, we would attempt to produce identical sterile environments, in which each set of experiments could be run.

Figure 7: Architecture 1 Breast Cancer Training Time Summary

Figure 8: Architecture 1 Breast Cancer Accuracy Summary

As summarized in Figure 9, native low-level Tensorflow APIs consistently trained faster than Keras with a Tensorflow backend on the same datasets. TensorFlow training time performance was more variable on the majority of datasets. Keras provided higher accuracy on the majority of datasets. Accuracy was negligibly variable across the MNIST and Fashion MNIST datasets for both frameworks. However, interestingly, there is observable variance in the cancer predictions- our most complex dataset. This in part is likely because our models had converged for MNIST and Fashion MNIST but had yet to do so for the cancer data, which would require a more finely-tuned model- not a model selected only for benchmarking purposes.

Training Time (s)	Keras		TensorFlow	
	Average	Variance	Average	Variance
MNIST	296.97	19.58	228.84	15.17
F-MNIST	295.20	11.74	219.95	15.82
Cancer	33.48	0.46	27.84	4.09

Figure 9: Summary of Architecture 1 Training Time Results

Accuracy (%)	Keras		TensorFlow	
	Average	Variance	Average	Variance
MNIST	99.05	0.00	97.72	0.00
F-MNIST	89.96	0.00	86.39	0.01
Cancer	71.37	0.05	73.77	0.11

Figure 10: Summary of Architecture 1 Accuracy Results

Figure 11: Architecture 1 Training Time Comparison

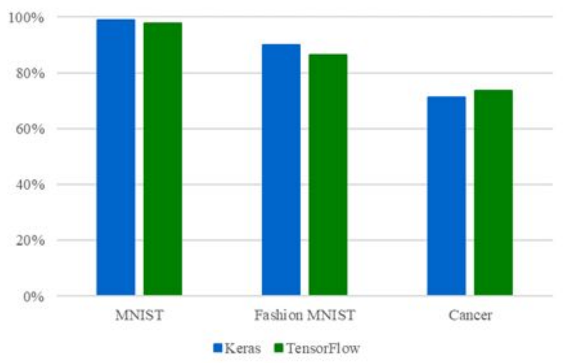

Figure 12: Architecture 1 Accuracy Comparison

C. Performance for Architecture 2

In the case of Architecture 2, native TensorFlow again faced greater variance in training time but did not outperform Keras with TensorFlow backend, on average, for Architecture 2. Compared to Architecture 1, for a majority of cases, training time increased for Architecture 2 by 2-3x. Accuracy for most use cases was also improved. Under this more complicated architecture, variance is proportionately tighter, despite variance in performance still being noticeable in training time. There was again negligible variance in predictive accuracy on MNIST and Fashion MNIST, but some on the breast cancer data set. This seems like a reasonable result due to the complexity of the breast cancer data set images. The low training time results of training on the breast cancer data set images was due to the low number of training instances in the data set relative to the MNIST and Fashion MNIST data sets. Had the number of breast cancer data images been the same as the number of MNIST and Fashion MNIST data images, we likely would have seen a significantly higher training time and potentially an exaggerated variance result, again due to the increased complexity of the larger, colored breast cancer images.

As can be observed in Figure 13, training time differences between Keras with TensorFlow backend and native TensorFlow were reduced under Architecture 2. Surprisingly, TensorFlow training time on Fashion MNIST took longer than training time on MNIST. We are unfortunately unable to rule out external factors influencing the Tensorflow vs Keras runs as a possible explanation for this unexpected result. This did not translate to improved accuracy, as compared to the Keras results.

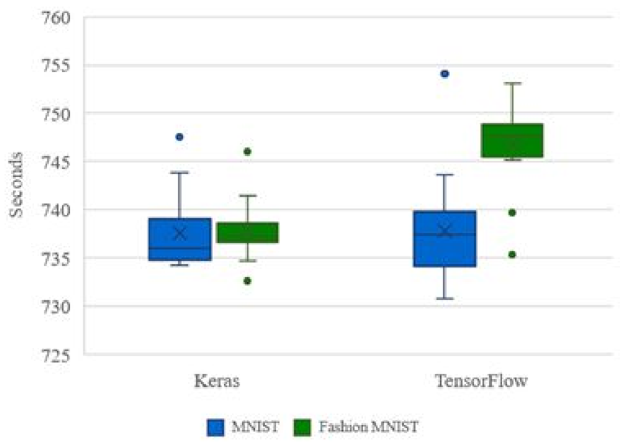

Figure 13: Architecture 2 MNIST/ Fashion MNIST Training Time

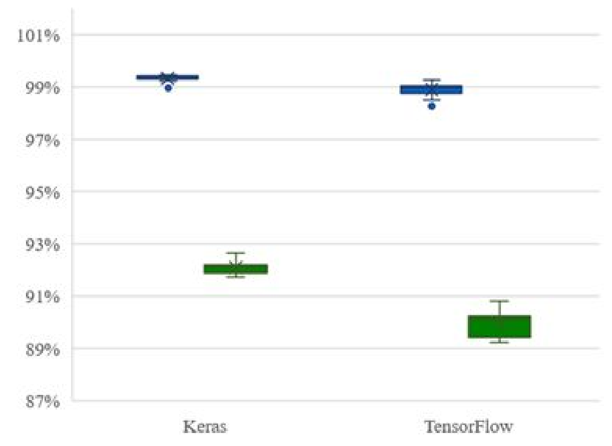

Figure 14: Architecture 2 MNIST/ Fashion MNIST Accuracy

When training Architecture 2 on the breast cancer dataset, TensorFlow again required longer training time. Keras had greater variance in training time, due to two outliers (Figures 15 and 16). We again suspect that the variances and outliers in our results are heavily impacted by the way in which Colab allocates resources on the backend to our instance.

The breast cancer dataset is again the only dataset of our three that presents variance in accuracy. We again suspect that this variance likely occurs due to the inability of Architecture 2 to reach convergence on the more complicated dataset.

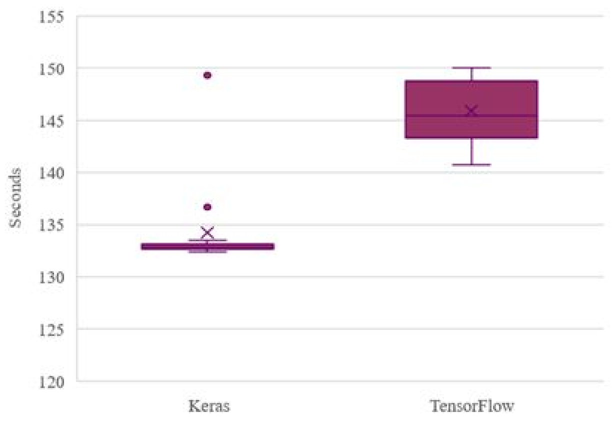

Figure 15: Architecture 2 Breast Cancer Training Time Summary

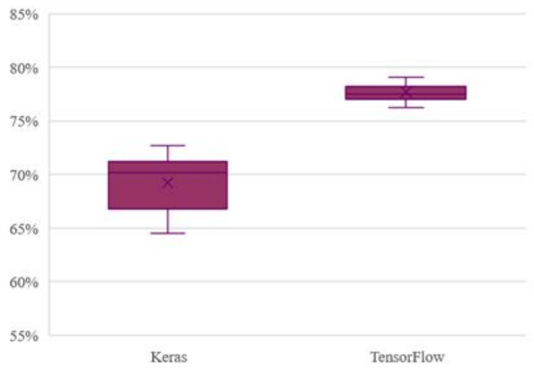

Figure 16: Architecture 2 Breast Cancer Accuracy Summary

Overall for Architecture 2- the deeper, more complex convolutional neural network of our two architectures- difference in performance between Keras and TensorFlow was minimized relative to the difference observed in experimentation with Architecture 1.

TensorFlow variance for training time was greater on MNIST and Fashion MNIST, while Keras training time variance was greater on the breast cancer dataset. Keras accuracy variance was also greater on the breast cancer dataset.

Training Time (s)	Keras		TensorFlow	
	Average	Variance	Average	Variance
MNIST	737.58	15.02	739.10	21.62
F-MNIST	737.91	9.82	746.82	21.67
Cancer	134.24	19.90	145.87	7.72

Figure 17: Summary of Architecture 2 Training Time Results

Accuracy (%)	Keras		TensorFlow	
	Average	Variance	Average	Variance
MNIST	99.32	0.00	98.90	0.00
F-MNIST	92.09	0.00	89.97	0.00
Cancer	69.25	0.07	77.69	0.01

Figure 18: Summary of Architecture 2 Accuracy Results

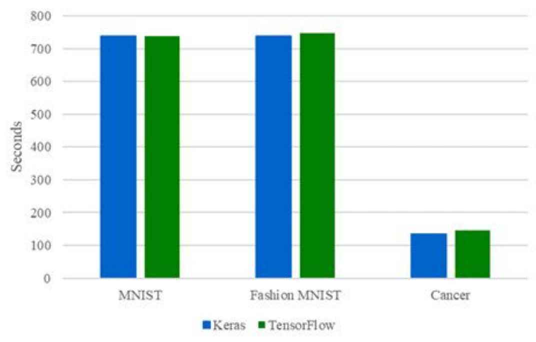

Figure 19: Architecture 2 Training Time Comparison

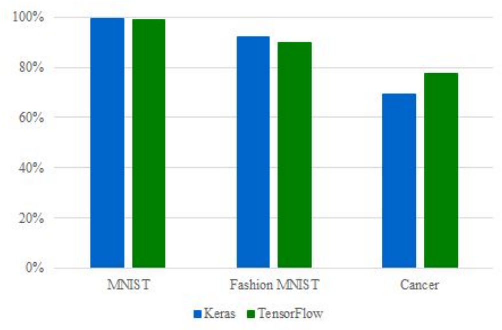

Figure 20: Architecture 2 Accuracy Comparison

D. Significance Testing

The empirical objective is to estimate whether there is significant difference between the training times or the accuracies of Keras and TensorFlow.

$H_0 : \mu_x = \mu_y$(*which can be rewritten* $H_0 : \mu_x - \mu_y = 0$), the test statistic is

$$t = \frac{\bar{x} - \bar{y}}{\sqrt{\frac{s_x^2}{n_x} + \frac{s_y^2}{n_y}}}$$

We are aiming to test to two hypothesis.

1) H_0: There is no significant difference between the training times of Keras and TensorFlow.

For MNIST using Architecture 1:

$$t_{1M} = \frac{296.97 - 228.84}{\sqrt{\frac{19.58}{15} + \frac{15.17}{15}}} = 44.76$$

For F-MNIST using Architecture 1:

$$t_{1F} = \frac{295.20 - 219.95}{\sqrt{\frac{11.74}{15} + \frac{15.82}{15}}} = 55.52$$

For cancer using Architecture 1:

$$t_{1C} = \frac{33.48 - 27.84}{\sqrt{\frac{0.46}{15} + \frac{4.09}{15}}} = 10.24$$

From the output above, we know the t-stat for all three dataset using the first architecture are greater than t* = 1.96. We have to reject the null hypothesis that is no difference between the training times of Keras and TensorFlow with 95% confidence.

For MNIST using Architecture 2:

$$t_{2M} = \left| \frac{737.58 - 739.1}{\sqrt{\frac{15.02}{15} + \frac{21.62}{15}}} \right| = 0.97$$

For F-MNIST using Architecture 2:

$$t_{2F} = \frac{737.91 - 746.82}{\sqrt{\frac{9.82}{15} + \frac{21.67}{15}}} = 6.15$$

For cancer using Architecture 2:

$$t_{2C} = \left| \frac{134.24 - 145.87}{\sqrt{\frac{19.90}{15} + \frac{7.72}{15}}} \right| = 8.57$$

From the output above, we know the t-stat for both F-MNIST and cancer dataset using the second architecture are greater than t* = 1.96. However, the t-stat for MNIST is less than 1.96 which means for that case we cannot reject the null hypothesis for MNIST using architecture 2, but for F-MNIST and cancer we reject the null that is no difference between the training times of Keras and TensorFlow with 95% confidence.

2) H_0: There is no significant difference between the accuracies of Keras and TensorFlow.

For MNIST using Architecture 1:

$$t_{1M} = \frac{99.05 - 97.72}{\sqrt{\frac{0.0}{15} + \frac{0.0}{15}}} = 1506.89$$

For F-MNIST using Architecture 1:

$$t_{1F} = \frac{89.96 - 86.39}{\sqrt{\frac{0.0}{15} + \frac{0.01}{15}}} = 1379.23$$

For cancer using Architecture 1:

$$t_{1C} = \frac{71.37 - 73.77}{\sqrt{\frac{0.05}{15} + \frac{0.11}{15}}} = 23.23$$

For MNIST using Architecture 2:

$$t_{2M} = \frac{99.32 - 98.90}{\sqrt{\frac{0.0}{15} + \frac{0.0}{15}}} = 552.56$$

For F-MNIST using Architecture 2:

$$t_{2F} = \frac{92.09 - 89.97}{\sqrt{\frac{9.0}{15}+\frac{0.0}{15}}} = 1502.57$$

For cancer using Architecture 2:

$$t_{2C} = \left| \frac{69.25 - 77.69}{\sqrt{\frac{0.07}{15}+\frac{0.01}{15}}} \right| = 115.57$$

From the output above, we know the t-stat for all three dataset using both architectures are greater than critical value t* = 1.96. Thus, we reject the null hypothesis that is no difference between the accuracies of Keras and TensorFlow with 95% confidence. This was surprising because the architecture of the model was the same across Keras and Tensorflow implementations. Because Keras was using TensorFlow as its backend, we expected the accuracy results of the Keras and Tensorflow implementations to be within a small margin of each other.

5 CONCLUSIONS AND FUTURE WORK

This paper presented a performance evaluation of two software frameworks designed for processing deep learning models on GPU settings. Two deep neural network architectures were implemented for three learning tasks i.e. handwritten digit recognition from the MNIST, clothes images from the Fashion MNIST, and breast histology images.

We observed that Keras with a TensorFlow backend has less variance in performance on the training time metric than native Tensorflow. Generally, Tensorflow could be expected to be faster than Keras. There was little variance in accuracy for either framework. When architecture complexity increased, differences in training time and accuracy between the two frameworks was reduced. When data complexity was increased, we observed more variance in our performance metrics for both frameworks.

We observed statistically significant differences between Keras and TensorFlow for all of our applications, with the exception of Architecture 2 on MNIST in both performance metrics of interest: training time and accuracy. We again suspect that this has more to do with Colab allocation of resources to our instance than it does with actual differences between the two frameworks.

In additional research, we would be interested in carrying out similar performance evaluations, including other frameworks (such as Pytorch, Theano, MXNet, amongst others) with more state-of-the-art deep learning architectures. We would also be interested in evaluating the additional frameworks on more complicated and even larger scale datasets. For instance, ImageNet or Google Open Images dataset with over 9M images, spanning 6000 categories.

For additional experimentation, we would be more critical about the resources we choose to train our models on in an attempt to evaluate core differences between frameworks, reducing the noise in our results from external/ unrelated factors. As previously mentioned, if given the opportunity for additional research to reproduce our experiments, given unlimited resources, we would attempt to produce identical sterile environments, in which each set of experiments could be run.

ACKNOWLEDGMENTS

We thank Professor Ian Foster, Professor Rick Stevens, Sam Nickolay, and many classmates from the Deep Learning Systems course for their valuable feedback on this work. We also thank Google for cloud credits.

REFERENCES

[1] S. Braun, LSTM Benchmarks for Deep Learning Frameworks”, *arXiv:1806.01818v1 [cs.LG]* 5 Jun, 2018.C

[2] C. Coleman, D. Narayanan, D. Kang, T. Zhao, J. Zhang, L. Nardi, P. Bailis, K. Olukotun, C. Ré, M. Zaharia, “Dawnbench: An end-to-end deep learning benchmark and competition”, NIPS ML Systems Workshop, 2017.

[3] K. Zhang, S. Alqahtani, M. Demirbas, “A Comparison of Distributed Machine Learning Platforms”, 26th IEEE International Conference on Computer Communication and Networks (ICCCN), 2017.

[4] Rubén D. Fonnegra, Bryan Blair, Gloria M. Díaz, "Performance comparison of deep learning frameworks in image classification problems using convolutional and recurrent networks", IEEE Colombian Conference on Communications and Computing (COLCOM), 2017

INDIVIDUAL CONTRIBUTIONS

Our team consisted of two students: Connie Fan and Tianchu Shu. We divided the coding required between the two of us (Connie prepared the TensorFlow code, Tianchu prepared the Keras code). After the code was written, we proceeded to divide additional tasks required for our experiments between the two of us (Connie was responsible for running the models, Tianchu was responsible for analysis of statistical significance of our results).

For presentation elements and final deliverables, we worked together to construct our poster and analysis for our paper. Finally, we both enjoyed the course and learned a lot throughout the quarter. Thanks for all of the help!