# Comparison and Evaluation of C# Implemented Convolutional Neural Networks

James Scruggs

MTSU Dept. of Computer Science

## ABSTRACT

Data science is steadily spanning into the field of software engineering. This research compares and evaluates the Microsoft ML.NET library to utilize machine learning in C#. In this project, we compare accuracy and runtime of a machine learning algorithm in C# with another in Python utilizing transfer learning to determine the effectiveness of ML.NET library. Both approaches will utilize the same architectures (Resnet-101, Inception-v3, and MobileNet) for each individual test. The task for each test is image recognition on two datasets comprised of vehicle images. Due to the data streaming abilities provided by the ML.NET library, we achieved faster runtime with the C# implementation while maintaining similar accuracy in both Python and C# implementations.

## INTRODUCTION

Machine learning can be difficult to integrate into many existing applications, as many software engineers are working in industries with specific tool stacks. With Python dominating the field of data science, other languages like C# have become less popular in the field of data science. Efforts have been made to extend the interoperability of C# with libraries like TensorFlow. For software engineers that specialize in Microsoft's .NET framework and companies that employ many applications utilizing this framework, Microsoft has developed a library called ML.NET that bridges the gap [1]. As the popularity of machine learning increases, questions surrounding the choice of framework that is being used have arisen. While Python is the most commonly utilized language, the effectiveness of others, like C#, have yet to be extensively explored in machine learning.

Building a deep neural network (DNN) is a computationally heavy process that is often aided with the use of the TensorFlow library for Python [7]. With DNN's being a computationally heavy process, time spent training the network can become a significant constraint for many software engineers. This can be aided with the use of transfer learning. Transfer learning is the process of taking a pretrained network and using it as the building block for a similar domain of data [5]. Utilizing weights from a pretrained network has shown to be an effective strategy for improving the training time on many DNN scenarios. ML.NET offers the ability of utilizing transfer learning to accomplish a variety of DNN applications [1]. ML.NET offers four popular DNN architectures for image recognition: Resnet-50, Resnet-101, MobileNetV2 and Inception-v3. Resnet-50/101 was designed by Microsoft [3] while Inception-v3 and MobileNetV2 was designed by Google [11, 10] [9].

## SPECIFIC AIM

The aim of this research is to compare and evaluate the ML.NET library against Python utilizing transfer learning on an image dataset containing images of vehicles in categories related to damage.

## METHODS

In order to evaluate the effectiveness of a convolutional neural network developed in C#, we utilized two machine learning libraries. For our Python implementation, we used the Keras Library. This library allows us to utilize transfer learning. For our C# implementation, we used Microsoft's ML.NET Library which utilizes transfer learning for its image classification algorithms. These libraries allow us to import the pretrained Resnet-101, Inception-v3, and MobileNet models. When ML.NET performs image classification, it obscures its image transformations [12] and then runs the transformed input features through the transferred architecture one time. This one time pass through the transferred architecture is known as the bottleneck phase [8]. This bottleneck phase can be similarly recreated within Python using Keras. Because ML.NET obscures its image transformations, we were unable to exactly reproduce the image transformations in Python. For simplicity, we decided to opt for the architecture preferred sizing and scaling of the images in Python.

We started by creating three C# programs differing only by the architecture used (Resnet-101, Inception-v3, and MobileNet) with the aforementioned, bottleneck design and image transformations. This step was repeated for the Python implementations. We set the batch size to 10, the number of epochs to 30, and the learning rate to 0.01 for all implementations. The training data was 70% of the total number images and the remaining 30% was the validation data. The images used in this evaluation consisted of three categories of vehicle damage: minor, moderate, and severe. The total number of images was 1,150. We ran each program three times on an Intel Core i7-3770K CPU @ 3.5GHz with 16 gigabytes of main memory; these algorithms did not utilize GPU resources. Validation accuracy and runtime was logged and analyzed after each run.

## RESULTS

In terms of validation accuracy, we found that Python was consistently superior shown in Fig. 01, Fig. 02, and Fig. 03. For the Python Inception-v3 implementation, validation accuracy averaged .659 in its best run while the C# inception-v3 implementation's best run yielded .635. The results for the MobileNet architecture were similar with the Python implementation yielding average validation accuracy of .671 while the C# implementation yielded .642. Lastly, the Python and C# Resnet-101 implementations shared the most similar validation accuracies with the Python Implementation averaging .664 and the C# implementation averaging .657.

While validation accuracy is a good indication of the model's ability to learn, we did not neglect to consider the models runtime. We found the most significant differences between the C# and Python implementations was the runtime. It can be shown in Fig. 04, Fig. 05, and Fig. 06 that the C# implementation is vastly superior for all architectures in terms of runtime. For the C# Inception-v3 implementation, the best runtime was 2 minutes and 43 seconds while the Python implementation was 4 minutes and 31 seconds. This trend continued with the C# Resnet-101 implementation yielding its best runtime at 6 minutes and 41 seconds while the Python implementation yielded 9 minutes and 53 seconds. The C# MobileNet implementation yielded a runtime of 42 seconds while the Python implementation yielded 4 minutes and 41 seconds.

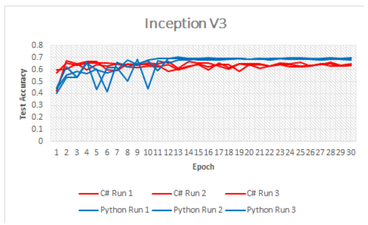


Fig. 01: Inception-v3 architecture validation accuracy across 30 epochs with 3 runs.

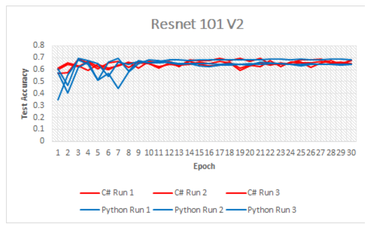


Fig. 02: Resnet 101 architecture validation accuracy across 30 epochs with 3 runs.

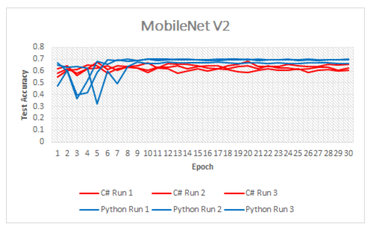


Fig. 03: MobileNet V2 architecture validation accuracy across 30 epochs with 3 runs.

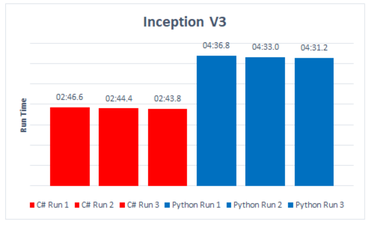


Fig. 04: Inception-v3 architecture runtime with 3 runs.

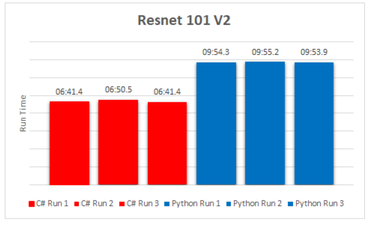


Fig. 05: Resnet 101 architecture runtime with 3 runs.

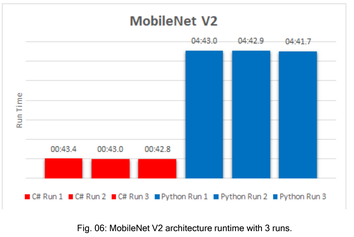


Fig. 06: MobileNet V2 architecture runtime with 3 runs.

## BACKGROUND

Liu, T. et al evaluated transfer learning to determine if it is a suitable approach to reduce training time while achieving satisfiable accuracy. The authors found that transfer learning is a satisfiable solution for most scenarios. The authors suggest that transfer learning could reduce the likelihood of overfitting [5]. Ahmed, Z. and others at Microsoft utilizes the strategy of transfer learning to bring DNN into their Microsoft ecosystem. For image recognition algorithms, ML.NET offers Resnet-101, Inception-v3, and MobileNet architectures. Microsoft has been utilizing the ideas of ML.NET for the last decade with their data scientists [1]. Ahmed, Z. et al compare ML.NET library against Sklearn (Python machine learning library) and H2O with classification and regression problems, in which the authors acknowledge the absence of DNN problems in this paper. The results showed ML.NET was superior in terms of CPU efficiency, and runtime [1]. One of the major contributions this paper offers is the concept of DataView, which allows ML.NET to consume data too large to store in main memory [1]. Kostelansky, E. et al have utilized the ML.NET library to estimate a train's travel time. The authors of this paper mention the advantages of using the ML.NET library for their application; however, the authors mention ML.NET lacks the tools necessary for data analysis [4].

Software engineering and data science are interwoven fields that are becoming disjoint. Because of this, software engineers may be less inclined to utilize machine learning for many of their applications. One could argue that this is caused by the extensive machine learning libraries offered in Python that are not offered in other programming languages [1]. For software engineers working in the .NET ecosystem, machine learning is more feasible now with the inception of ML.NET.

## REFERENCES

[1] Ahmed, Z., Amizadeh, S., Bilenko, M., Carr, R., Chin, W.-S., Dekel, Y., Dupre, X., Eksarevskiy, V., Filipi, S., Finley, T., and et al.,. Machine learning at microsoft with ml.net. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining, KDD '19, page 2448-2458, New York, NY, USA, 2019. Association for Computing Machinery.

[2] Chen, E., Wu, X., Wang, C., and Du, Y. Application of improved convolutional neural network in image classification. In 2019 International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI), pages 109-113, Nov 2019.

[3] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770-778, June 2016.

[4] Kostelansky, E., Krsak, E., and Kello, T. Estimate train driving time with artifical intelligence. In 2019 International Conference on Information and Digital Technologies (IDT), pages 237-244, June 2019.

[5] Liu, T., Alibhai, S., Wang, J., Liu, Q., He, X., and Wu, C. Exploring transfer learning to reduce training overhead of hpc data in machine learning. In 2019 IEEE International Conference on Networking, Architecture and Storage (NAS), pages 1-7, Aug 2019.

[6] Liu, X., Zhou, J., and Qian, H. Comparison and evaluation of activation functions in term of gradient instability in deep neural networks. In 2019 Chinese Control And Decision Conference (CCDC), pages 3966-3971, June 2019.

[7] Peker, M. Comparison of tensorflow object detection networks for licence plate localization. In 2019 1st Global Power, Energy and Communication Conference (GPECOM), pages 101-105, June 2019.

[8] Quintanilla, L. Tutorial: Automated visual inspection using transfer learning - ml.net, Dec 2019.

[9] Sai Sundar, K. V., Bonta, L. R., Reddy B., A. K., Baruah, P. K., and Sankara, S. S. Evaluating training time of inception-v3 and resnet-50,101 models using tensorflow across cpu and gpu. In 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), pages 1964-1968, March 2018.

[10] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L. Mobilenetv2: Inverted residuals and linear bottlenecks. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4510-4520, June 2018.

[11] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2818-2826, June 2016.

[12] Torre, C. D. I. Training image classification/recognition models based on deep learning & transfer learning with ml.net, Sep 2019.

## CONTACT / ACKNOWLEDGEMENTS

James Scruggs: jms2dv@mtmail.mtsu.edu

Faculty Advisor: Dr. Suk Seo