Data Processing and Machine Learning in the Cloud

Behzad Javaheri

Introduction: the overall goal of this work was to implement parallelisation and scalability in the cloud with Spark and TensorFlow. To achieve this aim we will parallelise the pre-processing stage and machine learning in the cloud and implement evaluation and analysis on the cloud performance. This report is structured to outline the results for section 1 on pre-processing and section 2 on machine learning in the cloud. This will be followed by a brief theoretical discussion to address questions raised in section 3.

1a) Herein raw measurement of throughput in images per second with different parameters including batch size, number and repetitions for reading directly from image files vs Tensor Flow Record files (TFR) provided. Our analysis shows that with all parameter sets, throughput was higher in TFR compared to image files and a direct relationship between batch size, number and repetitions and throughput was observed. The only exception was for image files experiencing a drop in throughput with higher batch size (Fig. 1).

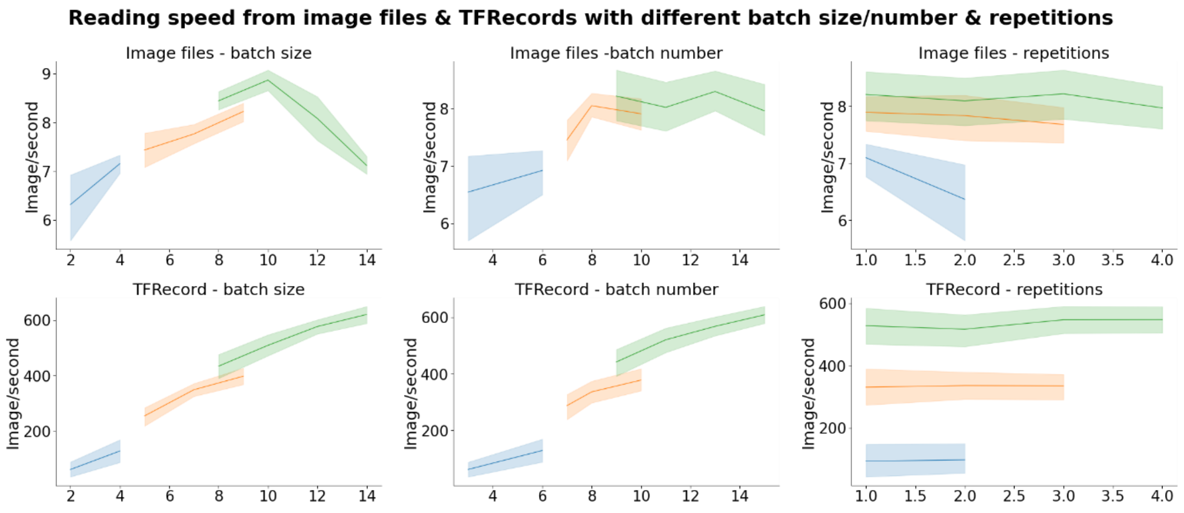

Figure 1. The comparative performance of throughput in images per second from TFR and image files with different parameter sets. Top panel) reading from the image files, and bottom panel) reading from TFR files.

1b) A linear regression over pre-processing and reading speed as a function of parameter values was performed. We find that throughput in images per second was higher (~ 70 folds) in TFR files compared to image files with direct and inverse correlation for TFR and image files with throughput, respectively (Fig.2).

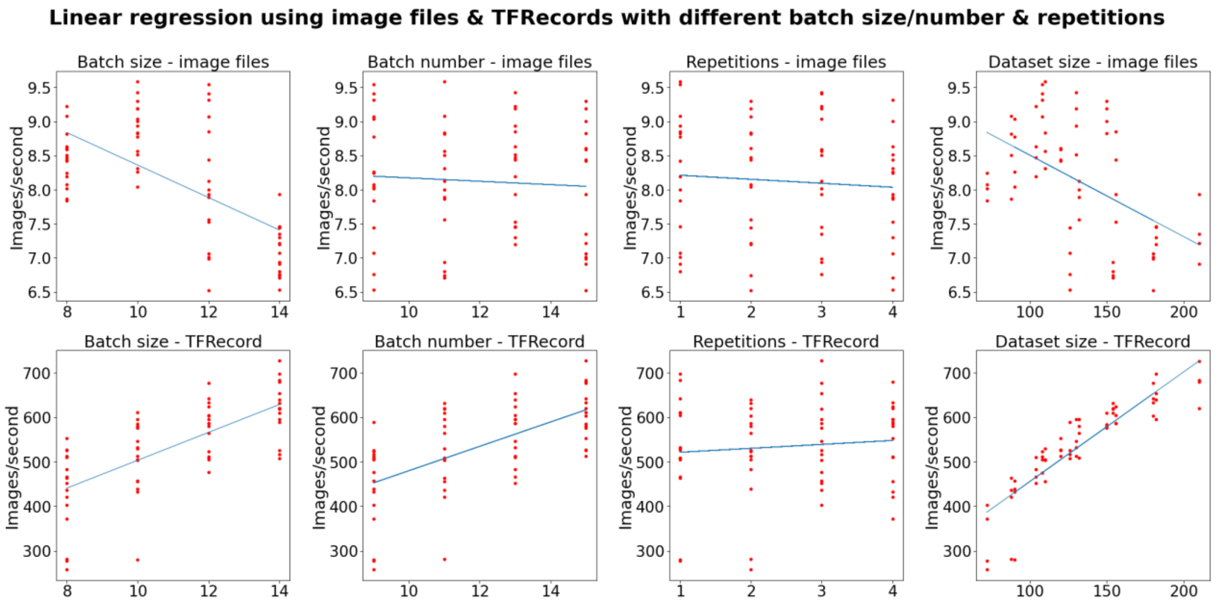

Figure 2. Throughput speed as a function of parameter values. Top panel) results from linear regression for reading from the image files and; Bottom panel) results from linear regression for reading from TFR files.

2b): Parallelising speed test with Spark without optimisation was performed. Analysis of cluster performance shows one initial peak after cluster creation. A period was left and marked as pre-submission for the cluster to stabilise before job submission. The second peak followed the job submission and continued for the duration of the job progression (~25 minutes). After which, completion declared, and CPU utilisation reduced to levels before submission. Average CPU use was ~10% and one CPU utilised significantly more than others at ~50%, whilst others remained close to ~5-10%. For network packets, a peak on incoming packets at the start of job processing observed at ~3.5 k/s and remained at ~0.3 k/s until job completion (Fig. 3A).

2c) Caching & parallelisation branches employed to enhance performance. Our data show that CPU utilisation reduced to ~10 minutes in response to cluster efficiency. Additionally, more workers seem to engage whilst the job was in progress. Average CPU utilisation was ~40% with 4 between 50-60% and the other 4 at ~20%. The approach in this step also resulted in a similar reduction in network packets indicating improvement in performance as a consequence of caching and parallelisation branches (Fig. 3B).

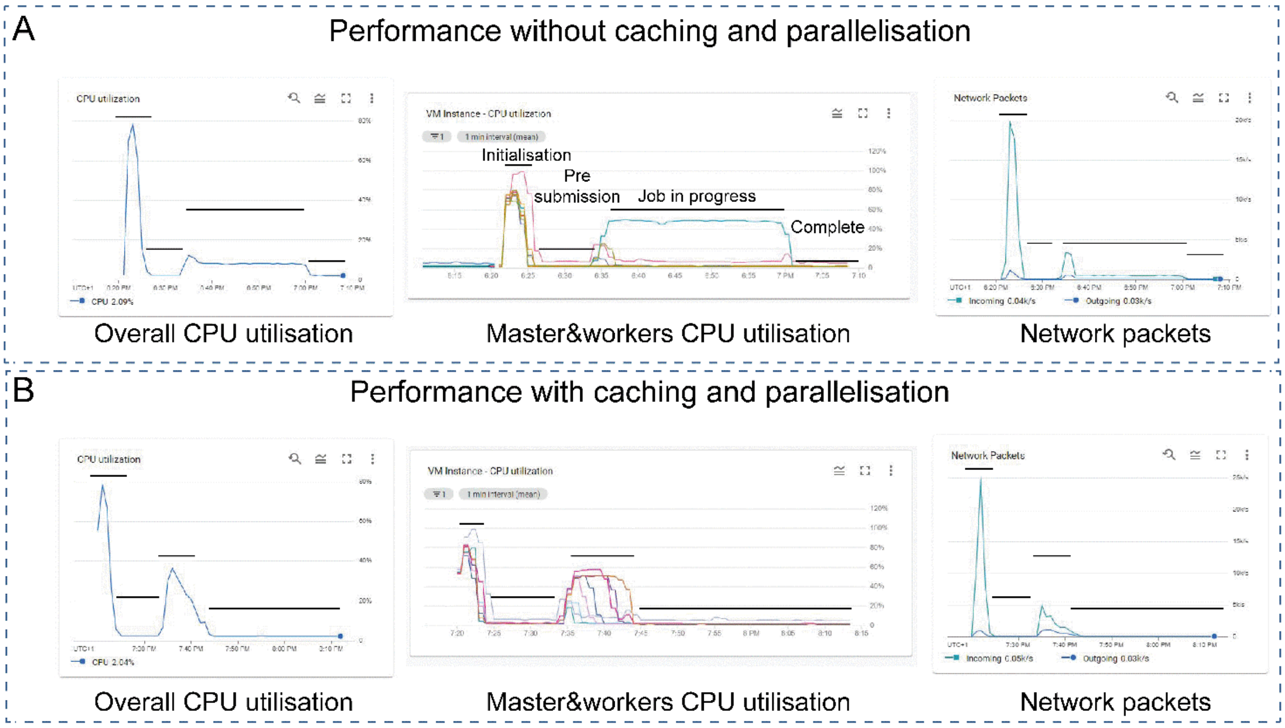

Figure 3. Cluster efficiency improves CPU utilisation and network performance. A) overall CPU utilisation, master and workers CPU utilisation and network packets when multiple speed tests are run in parallel using Spark. B) Similar to A but with cluster efficiency implemented by caching and parallelisation.

2d) Correlation coefficient data from linear regression analysis describe the strength and direction of a relationship between parameters and throughput. Our data indicate that where efficiency not implemented except for repetition (0.558) the correlation between other parameters and throughput was minor, negative for batch and data size and positive for batch number. The correlation of TFR in the same strategy with throughput was all positive and stronger. With the implementation of cluster efficiency, the correlation between parameters and throughput in the image dataset was positive and increased. A similar trend for TFR observed except for repetition where correlation decreased with cluster efficiency implementation (Table 1).

Without caching and parallelisation			**With caching and parallelisation**		
Type	**Parameter**	**R value**	**Type**	**Parameter**	**R value**
Images	Batch size	-0.155	Images	Batch size	0.313
Images	Batch no	0.157	Images	Batch no	0.094
Images	Repetition	0.558	Images	Repetition	0.198
Images	Data size	-0.019	Images	Data size	0.249
TFR	Batch size	0.293	TFR	Batch size	0.389
TFR	Batch no	0.262	TFR	Batch no	0.330
TFR	Repetition	0.475	TFR	Repetition	0.313
TFR	Data size	0.325	TFR	Data size	0.424

Table 1. Correlation coefficients of linear regression of multiple parallel speed tests in Spark with/out efficiency.

Figures 4 and 5 re-iterate these observations that with cluster efficiency the relationship between parameters and throughput enhanced except for repetition in the TFR file (Figs. 4-5).

Other alternative approaches can be considered. For example, approximation for throughput image per second can be performed using alternative functions including interpolation results and polynomials.

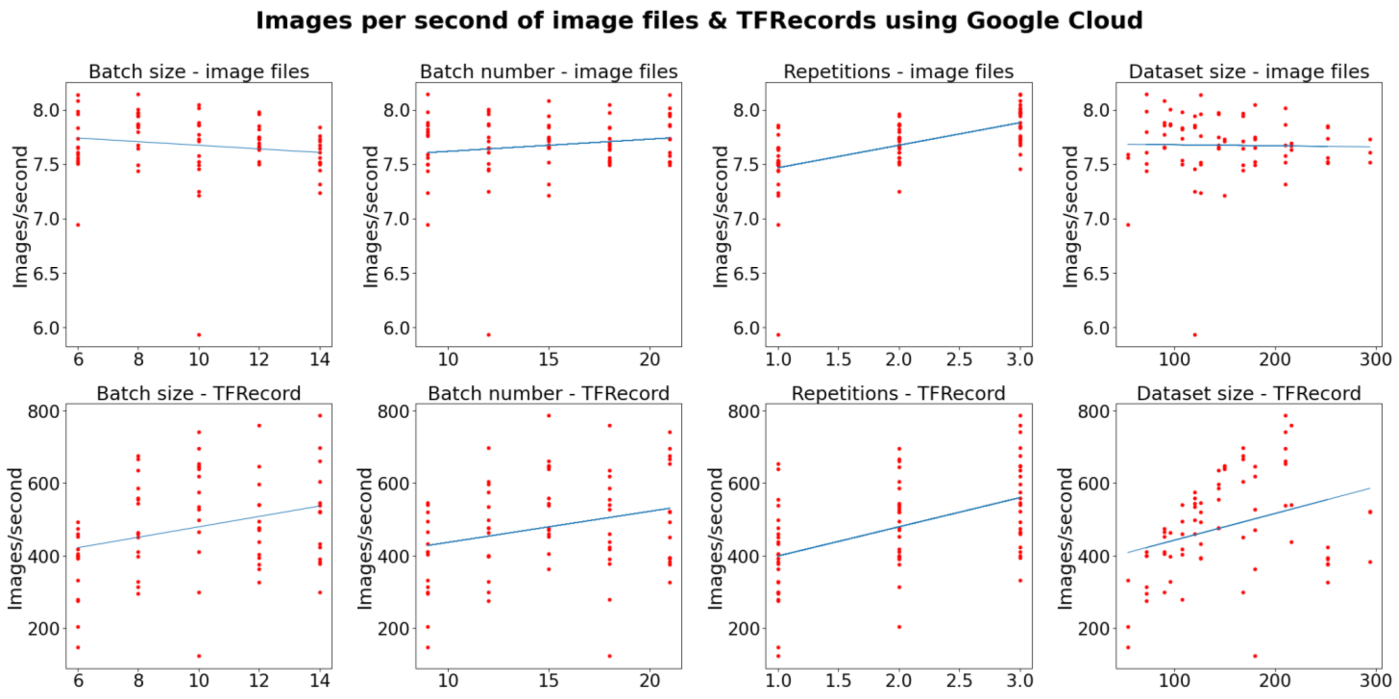

Figure 4. Linear regression of the output of multiple speed tests run in parallel using Spark. Top panel) results from linear regression for the image files and; Bottom panel) results from linear regression for TFR.

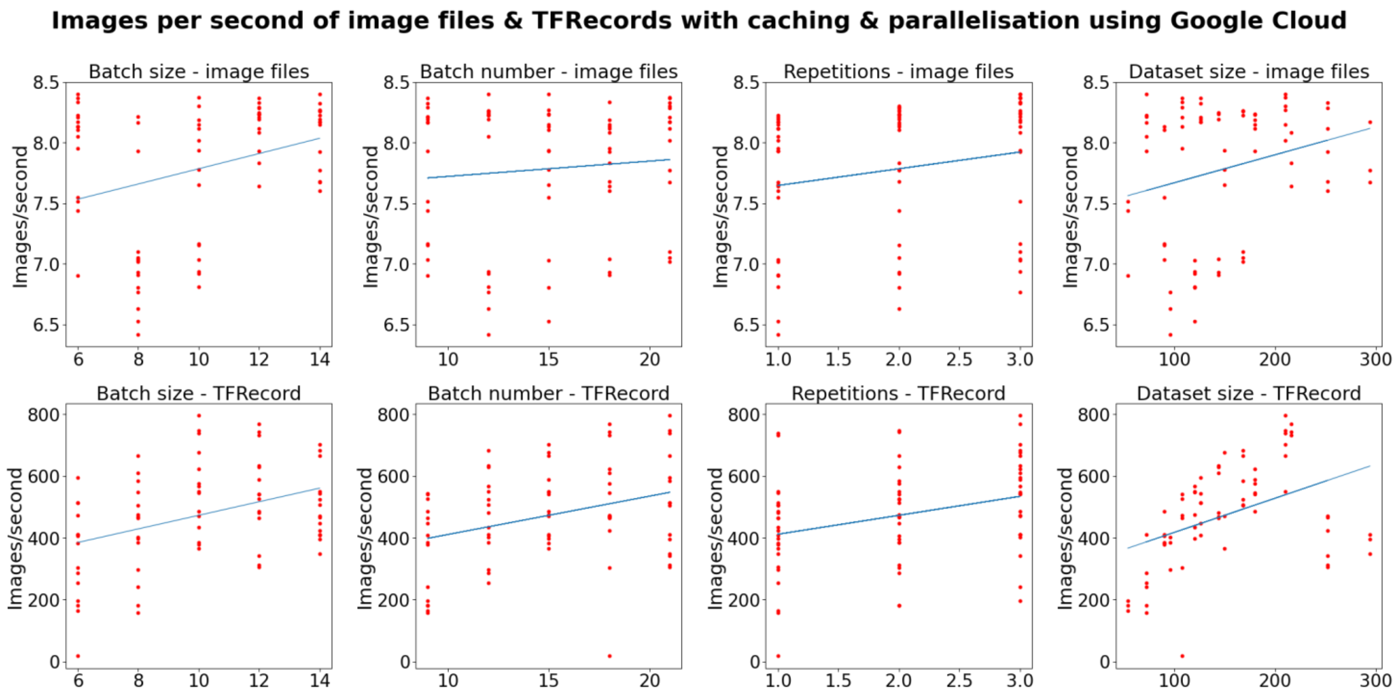

Figure 5. Linear regression of the output of multiple speed tests run in parallel using Spark with caching and parallelisation to improve cluster efficiency. Top panel) results from linear regression for the image files and; Bottom panel) results from linear regression for TFR files.

3ci) The use of Spark in the most standard applications include those employed during lab sessions to those employed in this work differ primarily as in the current work parallelisation over a cluster rather than a single node was performed. Furthermore, herein mapping was performed over partitions resulting in reduced computation time. To exploit the advantage of partitioning workers were employed for parallelised computation. The goal in this work was to achieve higher performance and lower computation time which can not be sufficiently adjusted using a single node. We have also, employed a larger dataset in this work and this has further confirmed the need for partitioning and use of workers.

3cii) Writing TFRecord files to the cloud with Spark was tested with two configurations of a) 4 machines with double resources (2 vCPUs, memory and disk: scenario 1) and b) 1 machine with 8 vCPUs (scenario 2). Analysis of cluster performance shows that the computation time for both scenarios was similar at ~4 minutes. However, with a single machine, ~27% of CPU was utilised at its peak whereas with 4 machines two of the workers utilised ~50%, one ~40% and the last worker ~20%. In addition, network packets for scenario 1 reached ~2k/s whereas for scenario 2 at ~1.3k/s. For scenario 1, incoming packets were slightly higher, and an opposite trend was observed for scenario 2. The number of disk operations was about 10 folds lower in scenario 2 compare to 1, where ~300 operations/s was observed for scenario 1, compared to ~30 operation/s. The read operations were higher than write operations in both scenarios.

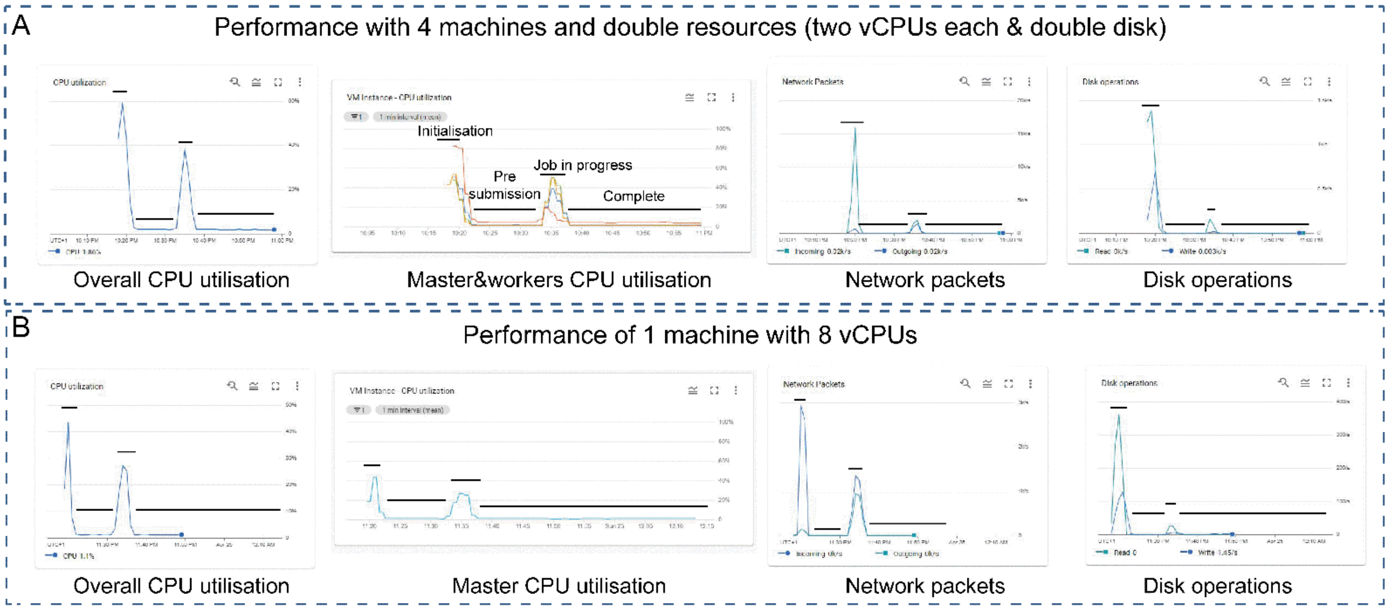

Figure 6. Cluster performance with two configurations. A) performance with 4 machine and double resources and B) performance with 1 machine and 8 vCPUs.

4b) Our data show average training accuracy of 89% with 0.615 loss for training and 74% accuracy with 2.19 loss for validation datasets. These show high accuracy for training and lower generalisation performance which may be enhanced by better cluster configuration and using optimal hyperparameters. Data augmentation may also enhance generalisation (Fig. 7)

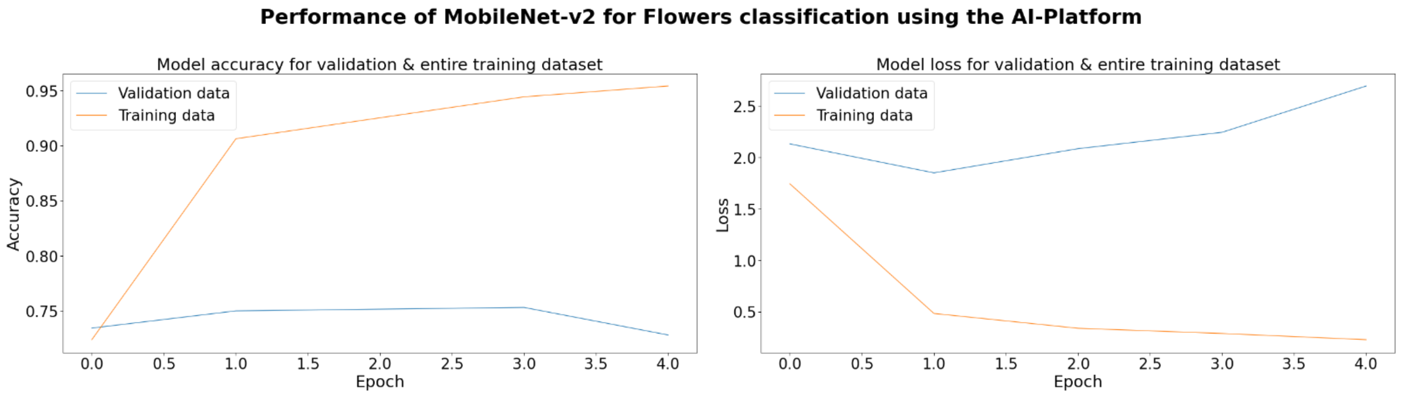

Figure 7. Performance of MobileNet-v2 for Flowers classification demonstrated by model accuracy and loss.

4c) Three distribution strategies with 3 varying batch sizes were initially considered. Amongst these, One Device and Mirrored strategies with batch sizes of 64, 256 and 512 were successfully implemented. Multi-worker Mirrored strategies did not, however, complete successfully due to cluster errors. Comparison of average model performance shows that One device strategy with batch size 256 produced the highest accuracy of 93.7 and 88.3% for training and validation datasets, respectively. The computation time for this was the highest at 20 minutes and 4 seconds. The lowest accuracy observed was for the mirrored strategy with 52.8 and 66.4% for training and validation. For One Device the training loss decreased with increasing batch size, in contrast, however, these increased for Mirrored strategy. For both strategies, the validation loss increased with increasing batch sizes (Table 2).

Strategy/batch size	Training loss	Training accuracy %	Validation loss	Validation accuracy %	Training time
One Device 64	0.452	85.3	0.400	86.6	16 min 33 s
One Device 256	0.306	93.7	0.682	88.3	20 min 4 s
One Device 512	0.341	93.6	0.816	86.6	17 min 3 s
Mirrored 64	0.558	82	0.467	85	12 min 32 s
Mirrored 256	0.535	83.8	0.473	85.4	14 min 3 s
Mirrored 512	1.593638	52.8	1.362	66.4	14 min 2 s

Table 2. Average model performance in Flowers classification with distributed learning strategies.

Analysis of model performance for each epoch re-iterates our earlier observation for both training (Fig. 8) and validation (Fig. 9) datasets.

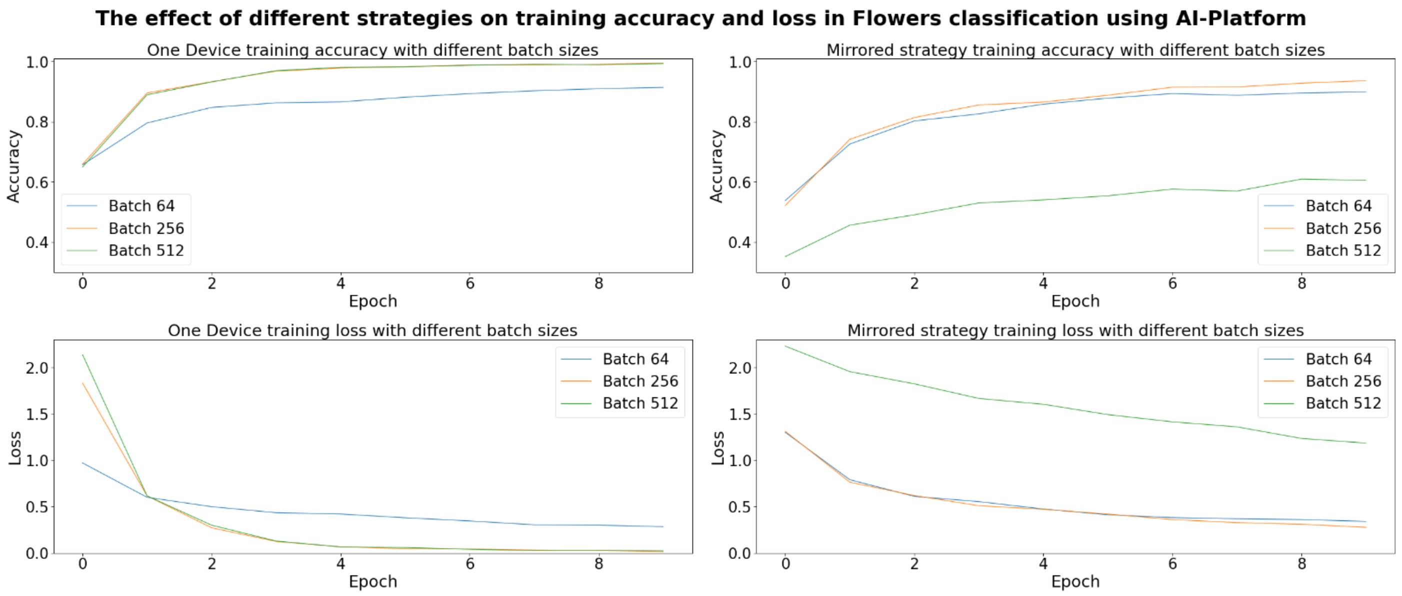

Figure 8. Performance of MobileNet-v2 for Flowers classification on the training dataset.

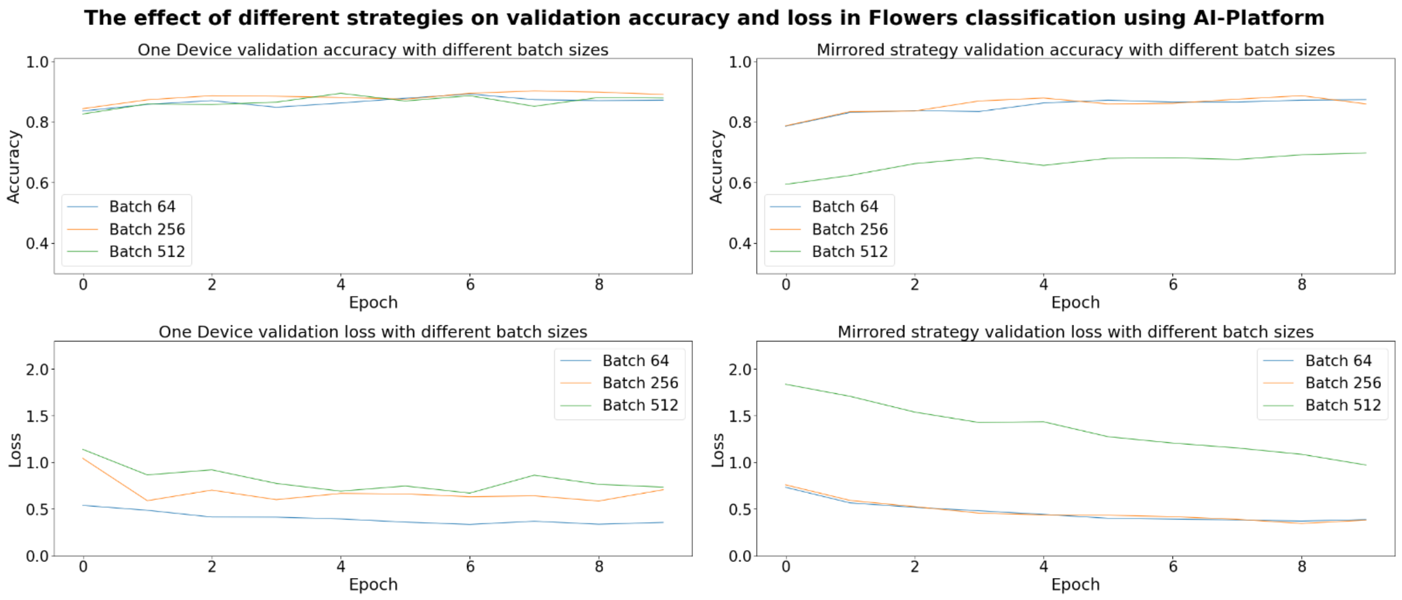

Figure 9. Performance of MobileNet-v2 for Flowers classification on the validation dataset.

Task 5:

5a) We have undertaken a journey from data pre-processing to running machine learning models in the cloud with different strategies. The work presented here specifically machine learning in the cloud is limited in scope as no structured strategy to identify optimal cloud configuration was performed. With real-world applications, this will lead to poor performance and higher costs [1].

Several strategies were undertaken in this work to examine their effects on model performance and computation time. These strategies included differing number of machines, number of workers, type of disk whether SSD or standard, vCPUs and implementation of distributed learning with varying batch sizes. Our data show that each of these elements affects performance as illustrated by accuracy and loss as well as computation time. This approach, however, is not systematic and does not test all possible configurations that might significantly impact overall model performance, runtime and cost. In addition, different cloud providers including Amazon, Microsoft and Google offer various cloud configurations. Therefore, a non-biased strategy to identify optimal configurations independent of these factors is essential.

To address this central issue, Alipourfard *et al*., (2017) devised CherryPick, an approach that aims to identify the lowest cost configuration with a runtime below a threshold. This approach is based on Bayesian Optimisation allowing CherryPick to implement adaptive search to estimate the optimal configurations. This adaptive search terminates once the expected improvement is below a defined threshold and after a defined number of configurations have been tried. Whilst CherryPick primarily considers the lowest running cost as the optimal selection criteria, the authors indicate that this trade-off between cost and quality of recommended configuration can be tuned by the users [1].

An additional optimisation approach is proposed by Smith *et al*., (2017) is to increase the batch size during training dynamically when the learning rate drops. The authors indicate that the benefits offered by learning rate decay during training can instead be achieved by enhancing batch size leading to a significant reduction in the number of parameter updates and consequently a reduction in training time [2]. For the One Device strategy, we observe increasing training accuracy and no significant effect on training time in contrast to the prediction made by Smith et al., (2017). Our data for Mirrored strategy however show a reduction in accuracy with increased batch size and increased training time.

The criteria for optimal configuration selection are important. Whilst overall cost is an important factor, this cannot be the determining factor in critical tasks where the highest possible accuracy is required. For example, for classification tasks in medical settings accuracy need to be the primary criteria.

5b) Many factors affect strategies for different application scenarios including batch, online and stream. In this work, we have adjusted some of these parameters and evaluated their effects on model performance, although no investigation of cloud costs in relation to these adjustments was performed.

Effective strategies need to tackle two central issues. First, we need to identify optimal machine learning algorithms suitable for our data type and the questions that need to be addressed. For example, linear regression may not be an ideal algorithm for multi-class classification and other algorithms such as Standard Vector Machines may perform better. Additionally, and following the selection of suitable algorithm, hyper-parameter tuning allows identifying parameters that produce the highest performance. These can be performed locally and only the best algorithm with ideal hyperparameters deployed into cloud. Alternatively, other approaches suggest dynamic alteration of parameters without hyper-parameter tuning [2].

Secondly, appropriate computation resources need to be identified. These may include an exhaustive search for cluster configuration to identify factors influencing computation including a search for the number of machines, vCPUs, memory, disk size and whether SSD or standard type, etc. Several approaches to identify these configurations have been developed. CherryPick [1], Micky [3], Ernst [4], PARIS [5], Arrow [6] and Scout [7] are amongst many proposals to identify optimal cluster configuration.

For each scenario, whether offline, online and stream processing a search to identify suitable algorithms to build models need to be performed. Whilst extensive hyperparameter tuning possible for offline data, may not be possible for online and stream data as computation, deployment and response in some cases need to be near-instant. As such it may be possible to employ re-enforcement learning specifically in simulation when prior data is not available. This would allow our model to have been trained with high accuracy without having prior data and therefore can be employed where data is generated every n second/minute requiring immediate processing. Additionally, resources and time permitting, a comparison of the most promising approaches to identify the best cluster configuration may allow the development of effective strategies for different application scenarios.

Number of words (excluding legends and references): 1989

Link to Colab:

https://colab.research.google.com/drive/1XzhjUUrXHIGwp4NkbeP67KyetCVZ1zP-?usp=sharing

References

[1] O. Alipourfard, H. H. Liu, J. Chen, S. Venkataraman, M. Yu, and M. Zhang, "Cherrypick: Adaptively unearthing the best cloud configurations for big data analytics," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 469-482.

[2] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le, "Don't decay the learning rate, increase the batch size," *arXiv preprint arXiv:1711.00489,* 2017.

[3] C.-J. Hsu, V. Nair, T. Menzies, and V. Freeh, "Micky: A cheaper alternative for selecting cloud instances," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, 2018: IEEE, pp. 409-416.

[4] S. Venkataraman, Z. Yang, M. Franklin, B. Recht, and I. Stoica, "Ernest: Efficient performance prediction for large-scale advanced analytics," in *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, 2016, pp. 363-378.

[5] N. J. Yadwadkar, B. Hariharan, J. E. Gonzalez, B. Smith, and R. H. Katz, "Selecting the best vm across multiple public clouds: A data-driven performance modeling approach," in *Proceedings of the 2017 Symposium on Cloud Computing*, 2017, pp. 452-465.

[6] C.-J. Hsu, V. Nair, V. W. Freeh, and T. Menzies, "Low-Level Augmented Bayesian Optimization for Finding the Best Cloud VM," *arXiv preprint arXiv:1712.10081*, 2017.

[7] C.-J. Hsu, V. Nair, T. Menzies, and V. W. Freeh, "Scout: An experienced guide to find the best cloud configuration," *arXiv preprint arXiv:1803.01296,* 2018.