Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.7664028

ACCELERATORS FOR PREDICTIVE MODELLING OF OMICS DATA

Deeba Khan
Ramaiah Institute of Technology, India
deebamajeed@yahoo.com

Seema S
Ramaiah Institute of Technology, India
seemas@msrit.edu

ABSTRACT

Predictive modeling of big-omics data using deep-learning techniques is gaining momentum due to its role in personalized treatment. Since modeling such data is both compute and data-intensive it is nearly impossible to achieve the task without accelerators. One of the affordable toolboxes that aids in modelling of the data in health-care is Google's Colaboratory. It is a cloud-based service that provides access to accelerators like GPUs and TPUs. In this work, we compared the performance of the accelerators hosted by Colaboratory to assess their advantages and shortcomings in the purview of clinical-science. The two models that we chose to train in this study are: a drug-response model and a cancer-subtype classifier. These predictive models help in the prognosis and diagnosis of carcinomas. We compared the performance of accelerators in terms of their training speed. We also compared the accuracy of the proposed models with similar methods. Results show that the models can be trained faster on the GPUs with smaller batches while TPUs achieve 5x-speedup for larger batches. The merits and pitfalls of the Colaboratory are also discussed at the end which may help the potential users. We conclude that performance of the accelerators with free/affordable options is sufficient for small research groups while they are insufficient for demanding problems that require ample processing power and memory.

Keywords: Deep Learning, Multi-omics data, GPU, TPU, AutoEncoder, Convolution Neural Network

INTRODUCTION

The 4.0 revolution in the field of bioinformatics has been a landmark for health-related sciences (*Big-Data-Artificial-Intelligence-and-Bioinformatics*, n.d.), giving rise to personalized medicines. This era of tailored therapy that is based on the patient's genome and its source is not possible without the support of supercomputers or accelerators which help in analyzing high dimensional multi-omics data supported by Deep Learning algorithms (DL). It is contended that Personalized Medicine will improve healthcare efficiency and reduce system costs (Marcon et al., 2018). Furthermore, it is also claimed that not only will personalized therapies help in accelerating the development of new treatments for terminal diseases but also, the drugs that were earlier considered ineffective might find new usages. These claims have fuelled the research both academically and commercially. Albeit, not all researchers and academicians can afford the toolbox needed for such experiments. For small setups and research groups, it may be tough to possess a powerful machine with multiple Graphics Processing Units (GPUs) for experiments. They need to consider under and overutilization of hardware resources, hardware deterioration, and failures. In addition, there are costs for maintenance, electricity and personnel to manage the hardware. Furthermore, providing each team member with a workstation attached to a sophisticated GPU is expensive. In these types of scenarios, the cloud-based platforms come to the rescue. Cloud solutions are increasingly popular because they provide on-demand infrastructure and reduce the need to manage and deploy hardware resources. On a pay-per-hour basis Amazon, Microsoft's Azure, and Google provide GPUs through their cloud services. Application-Specific Integrated Circuits (ASICs) like Tensor Processing Units (TPUs) along with deep learning runtime are also provided by Google. These cloud-based accelerators are easily available and free/affordable (Carneiro et al., 2018).

Based on the aforementioned scope, Google has established Colaboratory (Colab), a cloud service for sharing artificial intelligence-related training and research (*Google Colab*.). This service, popularly known as Colab is free to use and is tied to a Google Drive account. It also has affordable-paid versions with higher RAM and better GPUs, TPUs than the free version. Additionally, the deep learning frameworks are ready-to-use both on GPUs and TPUs. In this work, we compare the different accelerators at disposal for experimenting on Colab. GPUs and ASICs like TPUs are massively parallel devices and the right candidates to perform predictive modeling in clinical domains and for many other demanding applications. We consider comparing the performance of accelerators for specific problems rather than generalizing them. Such use cases may help researchers in a similar field to get a general idea and feasibility of using the cloud-based platform for their problems.

The primary goal of this paper is to study the feasibility of using Colab based accelerators for domain specific-deep learning applications. We experiment with clinical applications based on multi-omics data. Though some articles have published bench-marking results of accelerators available on Colab, they are broader thus undermining the real use cases. To accomplish the primary objective, we implemented two deep learning applications that exploit multi-omics data and need accelerators. These applications form the test-bed to assess the suitability of cloud-based accelerators. It can be seen that accelerator-based models are up to 10x faster than multiprocessor systems available on Colab. Finally, we go over some of Colaboratory's advantages and disadvantages, which may be helpful to new users.

Deep Learning and Omics data

Many researchers from diverse domains are adopting DL techniques for predictive analysis. In the clinical domain, both bio-medical and multi-omics data are being used to train the models. Training models using omics data is not straightforward due to its "big p small n problem" (Diao & Vidyashankar, 2013). Nevertheless, for these scenarios, DL methods have shown better performance and efficiency (Martorell-Marugán et al., 2019). For better clinical outcomes DL methods integrate different layers of omics data effortlessly allowing researchers to take a major leap towards personalized therapies, classification and early diagnosis of diseases, biomarker discovery and drug response prediction. These applications contribute to the much-talked-about “precision-medicine” which is gaining momentum. The different DL techniques that are being exploited to build models for the above applications include Convolution Neural Network (CNN), Graph-Convolution Neural-Network (GCN), Auto-encoders(AE) among many other methods. Key technical hitches encountered when using these DL techniques are demand for high-end processors for good performance and energy requirements to exploit this hardware. Since we are in the era of green computing we need to keep in mind the carbon footprints too. The aforementioned DL methods are found to have high accuracy index. For timely inferences and continuous training and retraining in the production environments, high-performance hardware is required. GPUs and TPUs are undisputed contenders for such tasks. Besides being massively parallel, such devices are energy-efficient (Carneiro et al., 2018).

Google Colaboratory

Colab is a Jupyter Notebook-based cloud service that promotes education and research related to machine learning and DL [6]. One can find several tutorials to learn its usage. Users have the option of using preconfigured Python 2 or 3 runtimes. It also supports deep learning libraries, such as Keras, Pytorch, Tensorflow along with Matplotlib for visualization. It can be configured to run R-based scripts (*R with Jupyter Notebook*, n.d.). The users are connected to Virtual Machine (VM) and the associated run-time through their Google account. The VM is disconnected after 12 hours resulting in loss of user’s data and configurations. Nevertheless, the user’s work on Jupyter Notebook is saved on user’s Google Drive (GD) account. The files can be transferred to and fro the connected hard disk and the user’s GD. File movement between GitHub and Colab’s hard disk is also gracefully supported. Colab also provides GPU/TPU-accelerated runtime that is fully configured with the necessary software for accelerated programming. Google Cloud platform (GCP) hosts the Colab services.

BACKGROUND

The most prevalent applications in biomedical research that are driven by omics data include the discovery of novel biomarkers for early cancer detection, therapeutic response, and labelling of patients. The high availability of omics datasets, mainly in oncology, for example, The Cancer Genome Atlas Program (TCGA) (Network et al., 2013), has allowed both DL and non-DL approaches to uncover new biomarkers. A promising study employed gene expression data to classify Breast Cancer(BC) observations from the Molecular Taxonomy of Breast Cancer International Consortium (METABRIC) or TCGA database as malignant or benign (Danaee et al., 2017). Additionally, using this strategy, a group of highly interrelated genes that were responsible for tumor growths were identified as biomarkers. Multiomics data from the TCGA has also been used to classify samples into different cancer types (Lyu & Haque, 2018). Si et al. employed an Auto-encoder to predict whether the subject is healthy or diseased based on methylation data (Si et al., 2016). Chatterjee et al. used CNN to categorize cancer types based on their methylation patterns, with promising results (Chatterjee et al., 2018). To classify liver cancer patients into distinct survival groups, Chaudhary et al. used multiple omics (RNA-seq, miRNA-seq, and methylation data) (Chaudhary et al., 2018). The authors used TCGA data to train their AE model. In a similar study, Olivier et al. utilized similar omics data from TCGA to stratify bladder cancer patients by their chances of surviving (Poirion et al., 2018). A survival and drug response prediction model for Breast Cancer developed by Malik et al., (Malik et al., 2021), is another DL-based precision oncology application. They proposed a robust, high-accuracy CNN-based model. The pan-cancer drug response model, a hybrid-graph-based Convolution Neural Network was proposed by Liu et al (Q. Liu et al., 2020).

The objective of our work is to compare the performance of a range of accelerators provided by Colab when used for demanding applications in health sciences. The applications are implemented using DL techniques along with omics data. This work gives the perspectives related to the limitations and benefits of accelerators. We see that the difficulty in implementing data-centric applications includes the processing speed and memory limitations of Colab. We train the multi-omics data to build predictive models using DL techniques. The first model is a classifier for BC subtyping that uses multi-omics profile for learning. The second model is a drug-response prediction model that suggests whether the patient is sensitive or resistant to a list of cancer drugs. These models have significance in the clinical field as they achieve the goal of personalized treatment. This work doesn't compare the performance of models built using different machine or deep learning techniques. Rather, we use the same DL technique and build model across different accelerators. Hence our results are mainly focused on accelerator performance than model performance. However, since the models' performance is an important aspect for AI systems, we have evaluated the performance of the proposed models too. The models we have proposed have a fairly better performance compared to similar methods and is discussed in results section of this article. In section 2 we describe the data and methods used for training the model. In section 3 we discuss performance of accelerators and models. In section 4 we present the advantages and pitfalls of Colab accelerators. The general workflow of the tasks carried out in the proposed work is as shown in Figure 1.

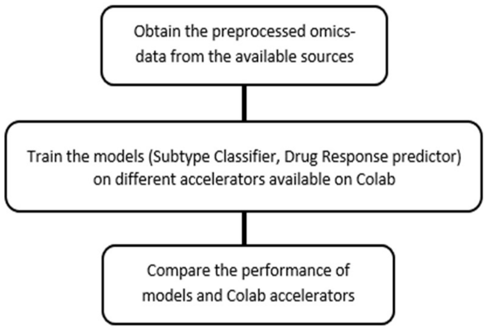

Obtain the preprocessed omics-data from the available sources
Train the models (Subtype Classifier, Drug Response predictor) on different accelerators available on Colab
Compare the performance of models and Colab accelerators

Figure 1. Steps carried out in benchmarking the performance

MATERIALS AND METHODS

Table 1 summarizes the accelerators' specifications (*Google TPUs*, n.d.; *Interl Processors*, n.d.; *NVIDIA*, n.d.) used in this study. We have chosen both Pro(paid) and free versions of the available hardware in our experiment. There is yet another paid version with better resources i.e. Pro+. We did not consider Pro+ in our study since it was not within the scope of our experiment. The processors include Intel's Xeon CPU, two types of GPUs from NVIDIA: Tesla K80 and P100, and finally Google's TPU. Their performance in terms of Giga(G) and Tera(T)

FLOPS are also listed. The RAM available at runtime is also listed. It can be observed that higher RAM is available for the Pro version. Table 2 gives a summary of the two models used during the performance analysis of accelerators. The first model we have developed is a classifier for BC subtyping that uses multi-omics profiles for learning. The second model is a drug-response prediction model that classifies the patient as sensitive or resistant to a list of cancer drugs.

Table 1. Description of the Colab-based accelerators used for evaluating the performance

Accelerators	**Model**	**Performance**	**Main Memory Support**
CPU_{free}	Hyper threaded Xeon Processors (1 core, 2 threads)	160 GFLOPS single precision	12 GB
GPU_{free}	NVIDIA Tesla K80 , with 24 GB, having 2496 CUDA cores	2.91 TFLOPS double precision	12 GB
TPU_{free}	TPUv2.0 with 8 cores/ TPUv1.0 with 8 cores	180 TFLOPS single precision	12 GB
CPU_{pro}	Hyper threaded Xeon Processors (1 core, 2 threads)	160 GFLOPS single precision	35 GB
GPU_{pro}	NVIDIA Tesla P100-PCIE-16GB with 3854 CUDA cores	4.7 TFLOPS double precision	25 GB
TPU_{pro}	TPUv2.0 with 8 cores	180 TFLOPS single precision	35 GB

Table 2. Summary of predictive models trained using different accelerators

Application	**Framework to support accelerators**	**DL methods**	**Number of features**	**Omics Integration**
BRCA Subtype Classification	Pytorch	Auto-encoder	46776 (without FS)	Early Integration
			1000(after FS)	

Pan-cancer Drug response prediction	Keras	CNN + GCN	36,181	Late Integration

Patient Classification based on Breast Cancer Subtype

One of the prediction problems in clinical setup is to classify the patient based on cancer subtype. Here we propose a classifier for identifying the BC subtype of the patient using PAM50. Though the popular routine for subtyping is based on immuno-histochemical tests (Nwosu & Piccolo, 2021), the intrinsic PAM50 subtype classification that is based on gene expression data is also widely accepted in clinical routines (Bastien et al., 2012) The omics data we considered for our experiment is from METABRIC (Curtis et al., 2012). It hosts clinical and omics profiles of patients and has PAM50-based subtype information for upto 80% of the samples. The four subtypes identified are: Her2+ enriched, Luminal A, Luminal B and Basal-like. BC subtype classification helps in choosing therapy for the patients that may be targeted therapy or chemotherapy. For instance, if a patient is classified into the Her2+ subtype category, one possible treatment strategy may be targeted therapy with Her2+ as the target. For a patient under the Basal-like category, treatment may be chemotherapy since there is no known target. More information on treatment strategies based on subtypes may be found in the literature (Lin et al., 2020). The omics data considered from METABRIC are the binary somatic mutation, copy number variation and gene expressions. The processed data is available at (*Moanna Data*, n.d.).

Implementation

The processed omics profiles were integrated to get the final 46776 features. The number of instances used for training was 1183. The classifier was developed using two early integration approaches. In the first approach, all the features from three omics sources were integrated and used for learning. In the second approach Feature Selection (FS) method, Chi-square was implemented to extract the top 1000 relevant features from the integrated profiles. Only these top-ranked features were used to train the model. The Pytorch framework was used to implement a partial Auto-encoder for learning i.e., only the encoder part of the method was used to derive optimal feature space using batch normalization. These encoded features served as input to the softmax layer for multiclass classification. This last layer uses weight decay and dropout for regularization. The approach is similar to the technique presented by Lin et al. [23]. The classifier predicts the patient's BC subtype. The identification of subtypes aids in picking the right therapy for the patients. The therapy may be targeted-therapy or chemotherapy. The model is evaluated with respect to accuracy and cost. The time taken for training on the different accelerators is also measured. The results are discussed in section 3.

Pan-cancer Drug Response Prediction

The next application considered is the drug-response model since it is gaining more popularity in the field of precision medicine. We trained the model for using pan-cancer cell line multi-omics data and molecular structure of the drugs. The model predicts the max IC50 concentration of the cancer drug required to inhibit cancer growth. Further, this model classifies the cell-line/patient as resistant or sensitive to a given drug based on the predicted IC50 value and threshold chosen. The details of the implementation are similar to the work proposed by Liu et al. (Q. Liu et al., 2020). We combined two data sources that were publicly available in this study: Genomics of Drug Sensitivity in Cancer (Iorio et al., 2016) and Cancer Cell Line Encyclopedia (https://sites.broadinstitute.org/ccle/, n.d.). Genomics of Drug Sensitivity in Cancer database provides IC50 values for up to 238 cancer drugs that include both chemotherapy and targeted drugs. Each IC50 value corresponds to a drug and a cancer cell line interaction pair. Cancer Cell Line Encyclopedia database provides genomic, transcriptomic and epigenomic profiles for more than a thousand cancer cell lines. These three omics data together have 36181 features. These profiles can be downloaded using the DepMap link (*DepMap*, n.d.). Natural log-transformed IC50 values of <cancer drugs, cancer cell lines> pair from the GDSC database reflect drug sensitivity profiles that help in measuring cancer drug response. We finally collected a dataset containing 107446 instances across 561 cancer cell lines. When combined with 238 drugs the resulting instances were a total of 133518. The total number of features of all the omics datasets combined is 36,181.

Implementation

This model is based on the late-integration Graph Convolution Network approach. The input to this framework is omics-specific subnetwork (i.e., a subnetwork of genomic data, transcriptomic data and epi-genomic data taken separately) that is used for mining information related to cell-lines and drugs on one hand and on the other hand, GCN takes into consideration the information of neighboring atoms in the drug-molecule while combining the features. Then the high-level features of drug and multiple omics data are merged and served to a 2-D CNN layer. To resolve the overfitting issue in the training process dropout and batch normalization were used after each convolutional layer. Adam optimizer was used for updating the parameters of the model. The final layer of the model was a sigmoid layer for prediction. Threshold of value 2 was used to classify the cell lines as resistant and sensitive. Cross-entropy was used as a loss function for this classifier. Keras library with the tensor-flow background was used to implement this GCN-based drug-response model. The results of this implementation are discussed in next section.

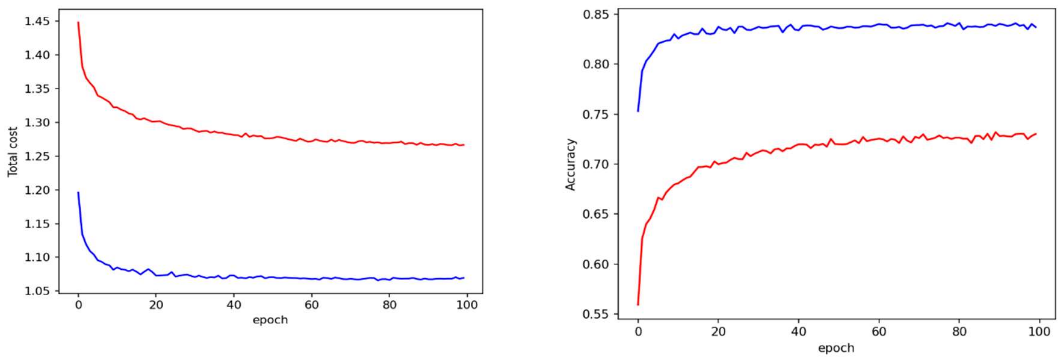
RESULTS

Subtype Classification BC

To train the subtype classification model 10-fold cross-validation was used. The accuracy of the model was persistent across all the accelerators. Figures 2 and 3 depict the performance of the model with respect to cost and accuracy after FS. Figure 2a shows the worst (1.29) and best performance (1.06) in terms of cost at different epochs with cross-validation. Figure 2b reflects the worst (0.72) and best performance (0.82) in terms of accuracy at different epochs with cross-validation. The overall accuracy of the model without FS was found to be 0.73. This reflects that, though not essential, choosing FS improves the performance of DL model. Figure 4 compares the performance of models for similar problem. Random Forest based model was implemented in-house while the details of other models like DeepMO (Lin et al., 2020)and MKL (Tao et al., 2019)can be found in the literature.

The model was trained on CPU, GPU and TPU with both free and Pro access. The training time of accelerators is as depicted in Figure 3. The training time with all features considered, batch size of 128 and 100 epochs is shown in Figure 4. With all features considered, the free versions of the accelerators GPU (K80) and CPU (Xeon) couldn't scale due to limited system memory of 12GB. GPU-pro showed the best performance for all the features considered. The time taken for training the model with reduced feature space after applying Chi-square is shown in Figure 5. The batch size and epoch used for training with feature selection are also 128 and 100 respectively. It was observed that varying the size of the batch affects the training time. Small batch size improves the performance of GPU while TPU has a good speedup rate with big batches. With reduced feature space, all the accelerators could train the model but with varying speeds.

The proposed model was trained with different batches: 64 and 128. The pro edition of TPU-v2 with a batch size of 128 had exceptional training speed with 5x and 2x improvement compared to CPU and GPU respectively. The model trained with a smaller batch size of 64 on GPU had a 2x gain compared to the larger batch size of 128 on the same GPU hardware. The batch inference time was approximately 65 seconds on the CPU and 30 seconds on TPU.



(a) (b)

Figure 2. The best and worst performance of the model (a) with respect to Cost. (b) with respect to Accuracy.

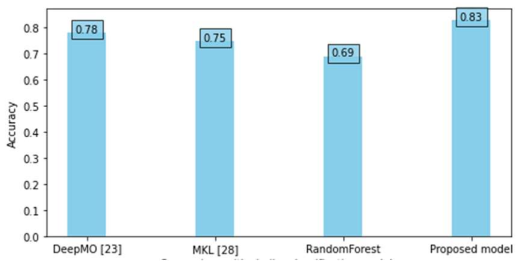


Figure 3. Comparison of state-of-the-art-models for Subtype Classification

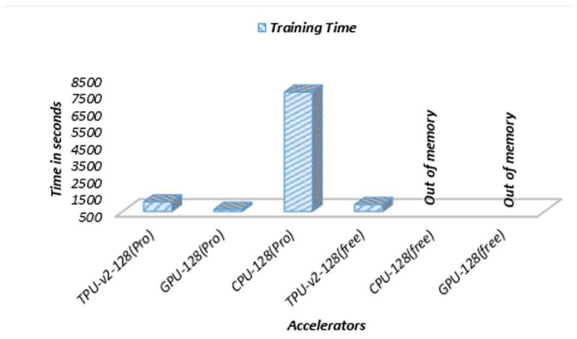


Figure 4. Time taken to train a classifier without feature selection. The batch size is 128

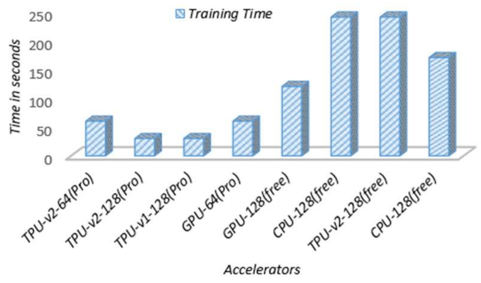


Figure 5. Time taken to train a classifer with feature selection. The batch sizes used are 64 and 128

Pan-cancer Drug Response Prediction Model

5-fold cross-validation was used to train the model and to improve its estimation power. Performance in all the cases i.e for all accelerators on which models were trained successfully remained constant. Figure 6 shows the operating characteristic of the model with an Area Under Curve (AUC) of 0.76. The accuracy of the model was found to be 0.83. The comparison of similar drug response models is depicted in Figure 8. The details of implementations of DeepCDR (Q. Liu et al., 2020) and tCNN (P. Liu et al., 2019) can be found in literature.

The proposed model was trained on CPU, GPU and TPU with both free and Pro access. The training time of accelerators is as depicted in Figure 7. The training time on different accelerators with a batch size of 128 for TPU/CPU and batch size of 16 on GPU is shown in Figure 8. The epoch in all cases is 50. Due to data enormity and limited memory, most of the accelerators failed to train the model. Even a small batch size of 16 could not be accommodated by the Pro version of GPU due to limited GPU memory. TPU with a large batch size of 128 achieved approximately 2.5x speed compared to CPU pro version and RAM support of 25GB. The batch inference time was approximately 0.81seconds on CPU and 0.42 seconds on TPU.
The batch inference time was approximately 0.81seconds on CPU and 0.42 seconds on TPU.

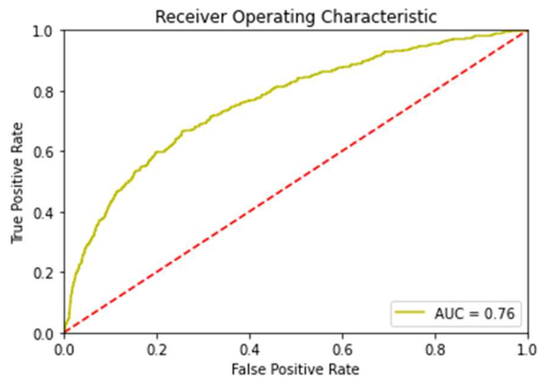


Figure 6. Performance evaluation of CNN based drug response model

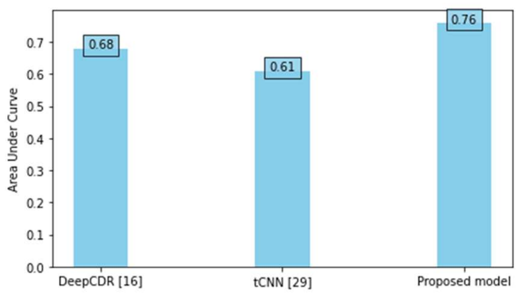


Figure 7. Comparison of state-of-the-art-models for Drug-effectiveness

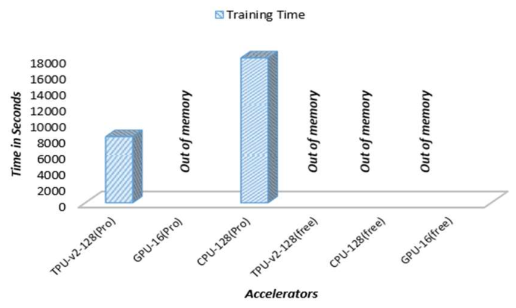


Figure 8. Time taken to build a Drug-Response prediction model on different acclerators

DISCUSSION

In this work, we compared the Colab accelerator performance with respect to clinical applications that are data intensive. The applications are related to predictive modelling that uses integrated omics data was to train two the models. The first model is a subtype classification model. The model was trained with and without incorporating the feature selection step. In both cases, the Pytorch-Tensorflow framework with a partial Auto-encoder was used to implement the training process. The model training failed in the case of CPU and GPU free versions due to limited memory when no feature selection was done. TPUv2.0 exhibited good training speed compared to other accelerators and achieved almost 5x performance.

The second model was trained using multi-omics and IC50 values of various cancer drugs. This model was used to envisage the sensitivity of the cell line/patient to a particular cancer-inhibiting molecule. The model was implemented using the Keras-Tensorflow framework with GCN as the training method. Due to data intensity, the model couldn't be trained on both free and pro versions of accelerators except TPUv2.0 and CPU with high RAM support. TPUs achieved 2.5x performance approximately. We can infer from the training time of both the models that TPU takes significantly less time to train than the GPU for the smaller batch sizes. But the Tensor Processing Units' performance is at par with GPU for larger batches. Our outcomes indicate that for a decent data workload with fairly good Colab resources the models can be trained as in the case of the first model. But as demand increases, Colab may fail to solve the purpose.

Accelerators hosted by Colab are sufficient for particular DL applications that use omics data but not always. Wherever possible, it is better to use the cloud services like Colab than a workstation with mid-end accelerators. On one hand, the resources at disposal by Colab are inadequate to solve all real-world applications with high computational and memory needs. While on the other hand, the DL frameworks are programmed for NVIDIA GPUs and Google TPUs. Also, the performance of the accelerators hosted by Colab may be sufficient for many researchers and students whose applications are less demanding. It is worth noting that the Internet set-up of Colab is swift, particularly while retrieving resources stored on Google Drive. This simplifies the manipulation of datasets using Colab. The accelerated runtime is fully configured which lessens the burden of novice users as they need not configure the accelerator drivers or compilers. The free-of-charge and affordable GPUs and TPUs are superior to several gaming GPUs, that democratize their access. In a situation of a research team with a small budget, this service may be obliging. Moreover, it is worth using the resources of Colab to decide on buying a dedicated computing system or availing cloud services for installing the applications. Also, it is easy to implement and test DL applications using Jupyter Notebooks. Apart from the aforementioned advantages, this cloud service has certain constraints that are discussed below. The main drawback is the time limit for accessing VMs and restriction on accelerator utility. All the resources along with VM are thrashed after twelve hours of their lifetime. The user needs to reconfigure the runtime from scratch again. However, the notebooks are preserved on GDrive and can be retrieved by the users. Google Drive interface is needed between Colab and the user's computer for the transfer of bulky datasets to be mined. In this case, the users need to be knowledgeable about the use of the APIs for file transfers between VMs and Drive. The drawback regarding this point is the transfer limits. Such limitations can be overridden by compressing the dataset via scripts. In conclusion, there is no assurance on the resource, especially the hardware, availability. Colab project may be scrapped by Google anytime and users may lose access.

Apart from accelerator comparison, the performance of our proposed models were compared with similar methods mentioned in the literature. Both the models exhibited fairly better compared to the existing methods.

CONCLUSION

Here, we presented the feasibility of using Colab for accelerating DL applications related to health care. It was observed that GPUs can train the model faster when the batch size is small while the TPUs are better choice for training model with large batches. Results reveal that it is worthy to train experimental models on Colab in the case of the research group or students who do not have access to better GPUs or ASICs in comparison with a K80, TPUv2.0 or P100. It is much easier to use Colab based services as they are straightforward. We also outlined the drawbacks of using Colab: there is a time limit on the VM and accelerator utilization, transferring data to/from Google Drive or GitHub. Moreover, the Colab hardware is not scalable. Also, it cannot solve bigger problems. A workaround is to have an API that can divide the problem in a way such that it can be run on multiple Colab instances. This solution will allow, multiple accelerators to be used with several Colab instances. The more apt solution is to go for paid services by Google, Amazon or Microsoft that host more powerful accelerators and RAM.

ACKNOWLEDGMENT

Conflict of Interest

"The authors of this publication declare there is no conflict of interest."

Funding Agency

"No Funding Information" heading.

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

REFERENCES

Bastien, R. R. L., Rodríguez-Lescure, Á., Ebbert, M. T. W., Prat, A., Munárriz, B., Rowe, L., Miller, P., Ruiz-Borrego, M., Anderson, D., Lyons, B., Álvarez, I., Dowell, T., Wall, D., Seguí, M. Á., Barley, L., Boucher, K. M., Alba, E., Pappas, L., Davis, C. A., … Martín, M. (2012). PAM50 breast cancer subtyping by RT-qPCR and concordance with standard clinical molecular markers. *BMC Medical Genomics*, *5*, 44. https://doi.org/10.1186/1755-8794-5-44

Big-data-artificial-intelligence-and-bioinformatics. (n.d.). https://www.telefonica.com/en/communication-room/blog/big-data-artificial-intelligence-and-bioinformatics-three-tools-that-save-lives/

Carneiro, T., Da Nobrega, R. V. M., Nepomuceno, T., Bian, G. Bin, De Albuquerque, V. H. C., & Filho, P. P. R. (2018). Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications. *IEEE Access*, *6*, 61677–61685. https://doi.org/10.1109/ACCESS.2018.2874767

Chatterjee, S., Iyer, A., Avva, S., Kollara, A., & Sankarasubbu, M. (2018). *Convolutional Neural Networks In Classifying Cancer Through DNA Methylation*.

Chaudhary, K., Poirion, O. B., Lu, L., & Garmire, L. X. (2018). Deep Learning-Based Multi-Omics Integration Robustly Predicts Survival in Liver Cancer. *Clinical Cancer Research : An Official Journal of the American Association for Cancer Research*, *24*(6), 1248–1259. https://doi.org/10.1158/1078-0432.CCR-17-0853

Curtis, C., Shah, S. P., Chin, S.-F., Turashvili, G., Rueda, O. M., Dunning, M. J., Speed, D., Lynch, A. G., Samarajiwa, S., Yuan, Y., Gräf, S., Ha, G., Haffari, G., Bashashati, A., Russell, R., McKinney, S., Caldas, C., Aparicio, S., Curtis†, C., … subgroup:, D. analysis. (2012). The genomic and transcriptomic architecture of 2,000 breast tumours reveals novel subgroups. *Nature*, *486*(7403), 346–352. https://doi.org/10.1038/nature10983

Danaee, P., Ghaeini, R., & Hendrix, D. A. (2017). A DEEP LEARNING APPROACH FOR CANCER DETECTION AND RELEVANT GENE IDENTIFICATION. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, *22*, 219–229. https://doi.org/10.1142/9789813207813_0022

DepMap. (n.d.). https://depmap.org/portal/

Diao, G., & Vidyashankar, A. N. (2013). Assessing genome-wide statistical significance for large p small n problems. *Genetics*, *194*(3), 781–783. https://doi.org/10.1534/genetics.113.150896

Google Colab. (n.d.). Retrieved February 10, 2022, from https://research.google.com/colaboratory/faq.html

Google TPUs. (n.d.). https://cloud.google.com/tpu

https://sites.broadinstitute.org/ccle/. (n.d.). *CCLE*. https://sites.broadinstitute.org/ccle/

Interl Processors. (n.d.). https://ark.intel.com/content/www/us/en/ark/products/series/125035/intel-xeon-w-processor.html

Iorio, F., Knijnenburg, T. A., Vis, D. J., Bignell, G. R., Menden, M. P., Schubert, M., Aben, N., Gonçalves, E., Barthorpe, S., Lightfoot, H., Cokelaer, T., Greninger, P., van Dyk, E., Chang, H., de Silva, H., Heyn, H., Deng, X., Egan, R. K., Liu, Q., … Garnett, M. J. (2016). A Landscape of Pharmacogenomic Interactions in Cancer. *Cell*, *166*(3), 740–754. https://doi.org/10.1016/j.cell.2016.06.017

Lin, Y., Zhang, W., Cao, H., Li, G., & Du, W. (2020). Classifying Breast Cancer Subtypes Using Deep Neural Networks Based on Multi-Omics Data. *Genes*, *11*(8), 888. https://doi.org/10.3390/genes11080888

Liu, P., Li, H., Li, S., & Leung, K.-S. (2019). Improving prediction of phenotypic drug response on cancer cell lines using deep convolutional network. *BMC Bioinformatics*, *20*(1), 408. https://doi.org/10.1186/s12859-019-2910-6

Liu, Q., Hu, Z., Jiang, R., & Zhou, M. (2020). DeepCDR: a hybrid graph convolutional network for predicting cancer drug response. *Bioinformatics (Oxford, England)*, *36*(Suppl_2), i911–i918. https://doi.org/10.1093/bioinformatics/btaa822

Lyu, B., & Haque, A. (2018). Deep Learning Based Tumor Type Classification Using Gene Expression Data. *BioRxiv*, 364323. https://doi.org/10.1101/364323

Malik, V., Kalakoti, Y., & Sundar, D. (2021). Deep learning assisted multi-omics integration for survival and drug-response prediction in breast cancer. *BMC Genomics*, *22*(1), 214. https://doi.org/10.1186/s12864-021-07524-2

Marcon, A. R., Bieber, M., & Caulfield, T. (2018). Representing a "revolution": how the popular press has portrayed personalized medicine. *Genetics in Medicine*, *20*(9), 950–956. https://doi.org/10.1038/gim.2017.217

Martorell-Marugán, J., Tabik, S., Benhammou, Y., del Val, C., Zwir, I., Herrera, F., & Carmona-Sáez, P. (2019). *Deep Learning in Omics Data Analysis and Precision Medicine.* (H. Husi (Ed.)). https://doi.org/10.15586/computationalbiology.2019.ch3

Moanna Data. (n.d.). https://zenodo.org/record/4326602#.Yi87QTXhW5d

Network, C. G. A. R., Weinstein, J. N., Collisson, E. A., Mills, G. B., Shaw, K. R. M., Ozenberger, B. A., Ellrott, K., Shmulevich, I., Sander, C., & Stuart, J. M. (2013). The Cancer Genome Atlas Pan-Cancer analysis project. *Nature Genetics*, *45*(10), 1113–1120. https://doi.org/10.1038/ng.2764

NVIDIA. (n.d.). https://www.nvidia.com/en-gb/data-center/

Nwosu, I. O., & Piccolo, S. R. (2021). A systematic review of datasets that can help elucidate relationships among gene expression, race, and immunohistochemistry-defined subtypes in breast cancer. *Cancer Biology & Therapy*, *22*(7–9), 417–429. https://doi.org/10.1080/15384047.2021.1953902

Poirion, O. B., Chaudhary, K., & Garmire, L. X. (2018). Deep Learning data integration for better risk stratification models of bladder cancer. *AMIA Joint Summits on Translational Science Proceedings. AMIA Joint Summits on Translational Science*, *2017*, 197–206. https://pubmed.ncbi.nlm.nih.gov/29888072

R with Jupyter Notebook. (n.d.). https://towardsdatascience.com/how-to-use-r-in-google-colab-b6e02d736497

Si, Z., Yu, H., & Ma, Z. (2016). Learning Deep Features for DNA Methylation Data Analysis. *IEEE Access*, *4*, 2732–2737. https://doi.org/10.1109/ACCESS.2016.2576598

Tao, M., Song, T., Du, W., Han, S., Zuo, C., Li, Y., Wang, Y., & Yang, Z. (2019). Classifying Breast Cancer Subtypes Using Multiple Kernel Learning Based on Omics Data. *Genes*, *10*(3), 200. https://doi.org/10.3390/genes10030200