Design Space Exploration of Deep Neural Network in Resource Constrained Devices

Anonymous Authors[1]

Abstract

A Deep Neural Network (DNN) is a type of artificial network consist of a large number of layers which is biologically-inspired in a way that mimicking human brain for learning and prediction. It can represent both liner and non-liner relationships. However, DNN is computationally heavy and expensive as it requires fast and parallel computations. The inference time and power consumption are closely related to the money as less time enables us to improve our work efficiency and less power consumption means fewer electricity bills. Normally, for a set of configuration, the more resource needed, the less inference time will be, and more power consumption will be generated. Thus, trade-off is desired between inference time and power consumption. In this paper, we highlight our work in exploring the configuration space in both hardware and compiler-level using sampling strategy. We show how Bayesian Optimization constructs an informative model in comparison to Random design in terms of exploring and exploiting different configuration regions. Finally, we utilized Polynomial Regression and Gradient-boosting regression to find out the relationships and interaction between configuration parameters given the outcomes, later to build a predictive model.

1. Introduction

The technological, industrial sectors become increasingly in favor of adopting Deep Neural Network as part of their overall systems due to its desirable features and predictive capabilities. However, DNN comes with a high price due to its architectural properties that requires time and consumes more power for performing extensive computations. Hardware configuration such as CPU frequency, and specificaly GPU and memory controller frequency, influences they way DNN performs in terms of both inference time and power consumption. Increasing configuration space such as maximum frequencies or batch-size results in decreasing inference time but at the cost of significantly increasing power consumption.

Initially, it was unclear in how to achieve the trade off between inference time and power consumption. In practical real world, it is unreasonable and unpractical for most applications, such as autonomous car, to utilize either maximum or minimum configurations for lowest inference time or lowest power consumption. Clearly, there are cases where having optimal or semi-optimal trade-off in order for autonomous cars, or mobile devices, to decide which next optimal configuration could be applied. For instance, autonomous car might need a configuration where it sacrifices slightly inference time in order to consume less power. Conversely, there are cases where autonomous cars needs to sacrifice power consumption to gain in much less inference time for prediction. Why selecting next configuration where there could a better choice? Meaning that, a configuration that is nearly cost-saving and possibly an optimal in the space region that could achieve both objectives, less power consumption or inference time in varying degrees which depends on circumstances. That is, we aim to explore configuration space in the deployment environment given the constrained hardware platform. Understanding and have a clear overview on how changing on configuration space effects DNNs performance is definitely one of our goals in this project. And, which configuration parameters most likely result in lowering both inference time and power consumption is another part of our goals as well.

We decided to perform our experiments on Nvidia Jetson TX1 due to its desirable, powerfull, and parallel computations. Additionally, We choosed 100 images from cifar10 dataset, and pre-trained both Resnet50 and VGG16 architectures from Keras, which is backed by Tensorflow, as DNN models. In this paper, we highlight some related works in this field. Subsequently, we discuss our sampling strategy to optimize both Power Consumption and Inference Time. Also, We discuss our approach in finding out the interaction between configuration parameters. We then discuss our

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

approach to predict both power consumption and inference time based on the obtained sampling results. Additionally, we show and discuss our experiment results and the different outcomes between both Bayesian Optimization and Random Sampling. Finally, we conclude our final thoughts on our sampling approach and discuss feature works.

2. Related Works

To explore the design space of a specified software system, various sampling designs are used to collect important information. [1] uses random sampling for multi-objective optimization. They proposed variations of a random search algorithm known as simulated annealing algorithm (SAA) for finding optimal sampling designs. [2] also summarizes Bayesian Optimization Approach from aspects of acquisition function and co-variance function choices. FREEZETHAW Bayesian Optimization mentioned in[3] talks about how to perform hyper-parameter tuning based on an assumption that training curves tend to follow an exponential decay. Using an information-theoretic decision framework, the algorithm can dynamically pause, resume, or create new training runs in order to rapidly find good hyper-parameter settings.

3. Data

We have incorporated Cifar10 image dataset as as our data input for DNN models. We only extracted first 100 images to perform prediction and classification. Since both Resnet50 and VGG16 models requires 224 224 dimensional image input, we transformed 100 images from Cifar10 dataset accordingly.

4. Methods

4.1. Dimension Space

We designed our dimensional configuration space on both hardware and compiler level. For hardware-level, we choose CPU frequency, GPU frequency, numbers of enabled cores, EMC frequency as a part of our configuration space. From compiler-level perspective, we decided to go with two options in Tensorflow, allowing memory growth and memory fraction per GPU. From program-level, we choose only batch size. As a result, exploration space of DNN can be defined as:

$$\Omega = O_1 \times O_2 \times O_3 \times ... \times O_n \tag{1}$$

where n is equal to 7, which is the maximum number of dimensional space.

4.2. Discretize Parameters

In order to conduct our experiments, we first needed to transform continuous parameter to discretized paramaters. For instance, during Guassian Process, the selection of parameters are knowingly continuous, and thus, we explicitly transformed them to the closest discretized value in the already available set of values.

4.3. Sampling Strategy

Evaluating and optimizing both Inference Time and Power Consumption is multi-Objectives problem. It takes long time and requires a huge amount of computations to evaluate - that is undoubtedly expensive. We specifically were looking for an optimization approach that evaluates multi-objectives function to shape an informative model while simultaneously performs as minimal iteration as possible. Our goal is to minimize both inference time and power consumption. These metrics can be regarded as functions of different configurations. So this is a optimization problem. Our goal is to tune hyper-parameters in order to find the optimal configurations formulated as follows:

$$c^* = argmin_{c \in \Omega}[f_1(c), f_2(c)] \tag{2}$$

After exploring various optimization techniques, we found that Bayesian Optimization a desirable and suitable candidate for our experiment. Bayesian Optimization is based on Gaussian Process to tune hyper-parameters, which uses previous observed parameters to make an assumption about unobserved configuration parameters. Therefore, we utilized an already existing Python library "GPflowOpt" [4] to perform multi-objective Bayesian Optimization. We also decided to use volume-based probability of improvement - as an acquisition function - to intelligently select next configuration parameters. That way, it looks for points where it can shape Pareto frontier, an optimal set of points in Pareto set of configurations that not only minimizes both objective functions, but also satisfies the optimal trade-off between both energy consumption and inference time.

4.4. Interaction of Dimensional Space

To further understand the interaction between different dimensional space of configurations, we use polynomial regression model to fit the data observed by Bayesian algorithm and formulate their relationship with various degree orders. Then we predict the performance of this built model and make a validation based on RMSE (Root Mean Square Error) with ground truth to find out the best polynomial degree which shows the most accurate interaction between configuration variables among other degrees.

4.5. Prediction

We would like to see how samples obtained from Bayesian Optimization could describe the data and further predict the outcomes based on the given configuration parameters. Moreover, how samples obtained by Bayesian Optimization approach differs from Random Sampling in terms of accurate prediction. We expect the outcomes to bear some high biases and variances due to its the small number of obtained samples. Thus, We decided to use Gradient-Boosting regression due to its rigorous features in controlling bias and variances in regression.

5. Experiments

As stated in the ***introduction*** section, our experiments are conducted on Nvidia Jetson TX1. TX1 is the world's first supercomputer on a module, Jetson TX1 is capable of delivering the performance and power efficiency needed for the latest visual computing applications. It's built around the revolutionary NVIDIA Maxwell architecture with 256 CUDA cores delivering over 1 TeraFLOPs of performance. 64-bit CPUs, 4K video endcode and decode capabilities, and a camera interface capable of 1400 MPix/s make this the best system for embedded deep learning, computer vision, graphics, and GPU computing

Handling values was a serious challenge in conducting experiments on VGG16 model due to memory utilization issues. Thus, We note that two of dimensional configuration space, both EMC frequency and Batch-size were more constrained in VGG16 model than in Resnet50. Therefore, it might possibly effected the outcomes as well as the comparison of sampling between Resnet50 and VGG16.

5.1. Sampling Result

We employed both Bayesian Optimization approach and random sampling to explore the configuration space and observe different outcomes. We initially performed 10 sampling using Latin-hyper cubes and run 30 iteration for Bayesian Optimization. On other hand, we run 40 iterations through Random sampling. As shown in figure 1 and figure 2, the sampling were performed on Resnet50 neural network architecture. Bayesian optimization formed Pareto frontier, optimal points, in Pareto set. Different colors indicates the dominance of points, as the purple colors dominate the other colors. Thus, we can confidently make a trade-off between power consumption and inference time. However, we could notice more scattered points in the random sampling, wheres, Gaussian process formed more a curved line which supposed to be the optimal set of configurations.

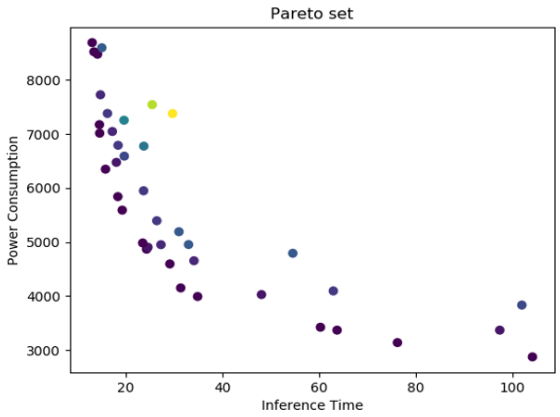

Figure 1. Pareto Set for Resnet50

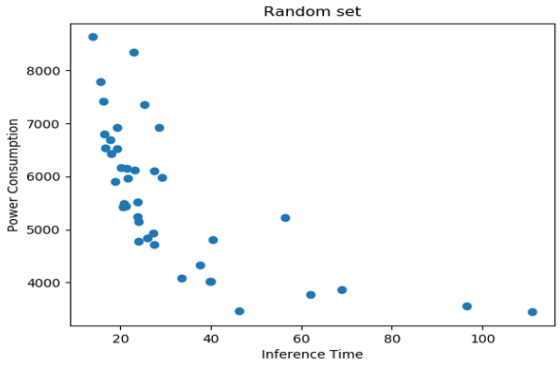

Figure 2. Random Set for Resnet50

Meanwhile, in figure 3 and figure 4 the samples where executed on VGG16 neural network architecture. The performance of Bayesian Optimization and Random sampling seems ambiguous and difficult to differentiate. Nonetheless, we notice that the Bayesian Optimization formed Pareto frontier in Pareto set. However, we were surprised to observe the Random design outcomes, as it resembles Pareto set of configurations. However, we can't confidently conclude whether Random sampling found similar set of points.

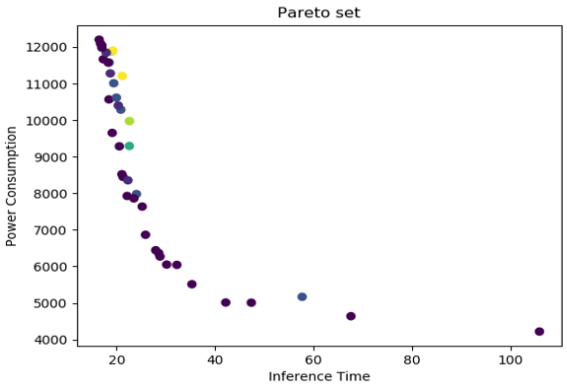

Figure 3. Pareto Set for VGG16

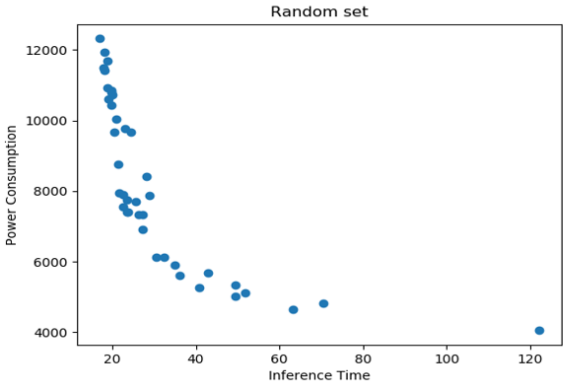

Figure 4. Random Set for VGG16

Clearly, figure 5 shows the difference in optimizing both functions, power consumption and inference time. Bayesian optimization approach were very helpful in minimizing both Power consumption and Inference Time. In contrary to Random samples, inference time dropped earlier in 27th iteration in Gaussian process. Additionally, we observe that the Bayesian Optimization proved to be efficient regarding the power consumption. As it shows in the figure 6, it exploited the region where it can minimize the power consumption for several times.

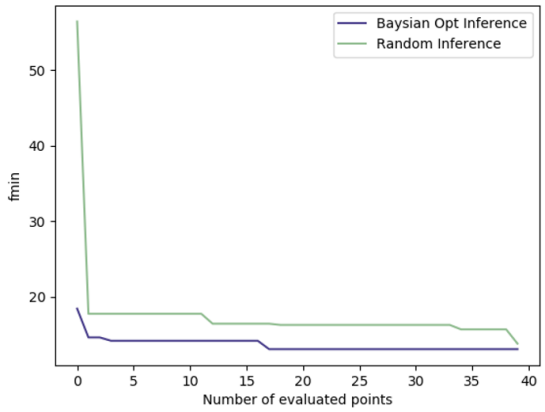

Figure 5. Inference time comparison between Bayesian and Random sampling for 40 iterations

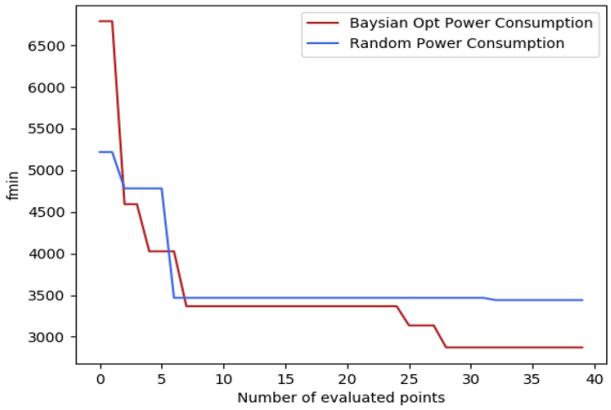

Figure 6. Power consumption comparison between Bayesian and Random sampling for 40 iterations

In figure 7, we observe closely in how Bayesian optimization did not minimize the inference time. Our speculation is that the regions where inference time could be exploited and further dropped but never minimized. However, we clearly see how random Sampling reached to the point higher than the point where Bayesian optimization achieved. On other side, we show in figure 8 how Bayesian optimization didn't optimize the power consumption as expected, which is in sharp contrast to Random Sampling. Our reasoning about this unexpected outcomes is that it could be a case where random sampling, as the name implies, happened to sample configuration parameters which resulted eventually in lower power consumption.

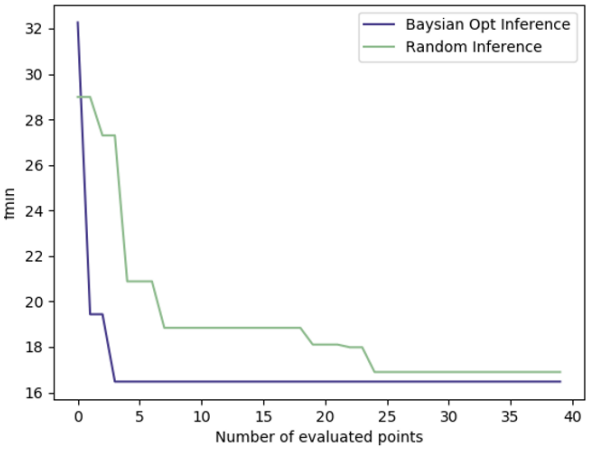

Figure 7. Power consumption comparison between Bayesian and Random sampling for 40 iterations

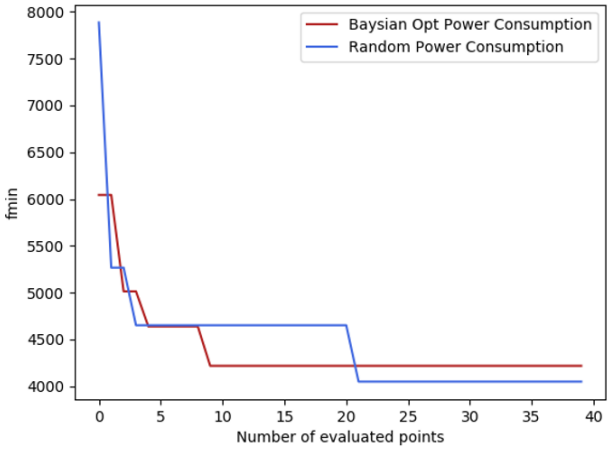

Figure 8. Power consumption comparison between Bayesian and Random sampling for 40 iterations

5.2. Polynomial Function

We have constructed polynomial function with the various degree. We empirically observed that the outcomes of polynomial degree 3, among 2 and 4, is best describing the interaction of parameters in regards to the outcomes, bother power consumption and Inference Time. We define the polynomial function as the following below for both inference and power consumption:

$$\begin{aligned} f_{inference} = {} & 58.29 + 1.7O_0 + 4.2O_2 + 18O_6 \\ & + 0.1O_4O_5O_6 - 1.4O_4O_6^2 + 18.1O_{5O}6 \\ & - 4.8O_6^2 + 93.2O_0^3 - 0.3O_1^2O_2 - O_1^2O_6 \\ & - 0.8O_1O_2^2 + 1.9O_5^3 \end{aligned} \tag{3}$$

$$\begin{aligned} f_{power} = {} & 2844.94 - 32.2O_1O_5^2 - 309.1O_5O_6 + 15.9O_4O_6 \\ & - 6.8O_1O_4O_5O_6 - 66.9O_1O_2^2 - 2622O_2^3 \\ & + 15.9O_4O_5O_6 + 31.6O_4O_6^2 \end{aligned} \tag{4}$$

where O_0 indicates CPU frequency, O_1 indicates number of disabled cores.... and O_7 indicates memory fraction per GPU. We clearly see the interaction of parameters and the coefficients in both polynomial functions for Inference Time and power consumption.

5.3. Prediction

As we stated in **Method** section, we hired Gradient-boosting regression to build a robust predictive model on samples generated by Bayesian Optimization and Random sampling. We note that we considered 80% of sample data as a training dataset, and the rest of data, which is 20%, as test dataset. On above figures, figure 9 and figure 10, are a comparison between Bayesian Optimization and Random sampling for predicting inference time taken from **Resnet50**. We calculated the RMSE for inference time predictions, and the outcomes as following: ***3.08 RMSE*** for samples obtained by Bayesian Optimization, and ***20.16 RMSE*** for random sampling.

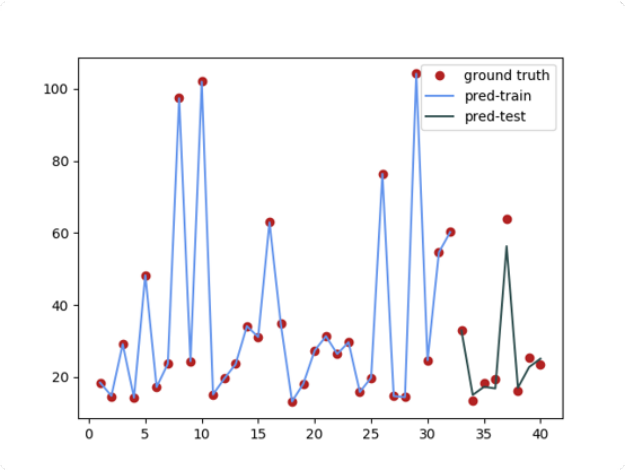

Figure 9. Resnet50 prediction on Bayesian samples for inference Time

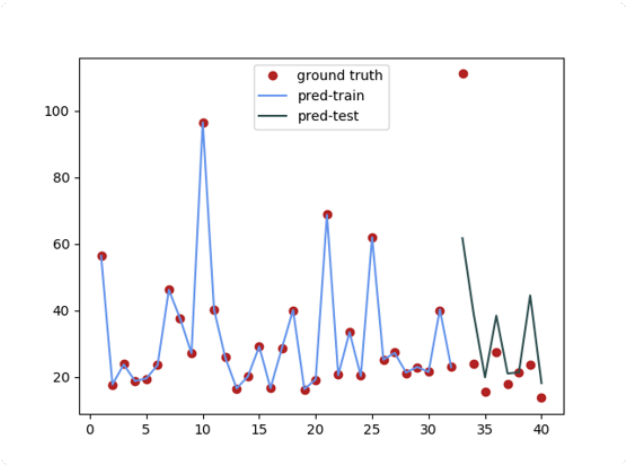

Figure 10. Resnet50 prediction on random samples for inference Time.

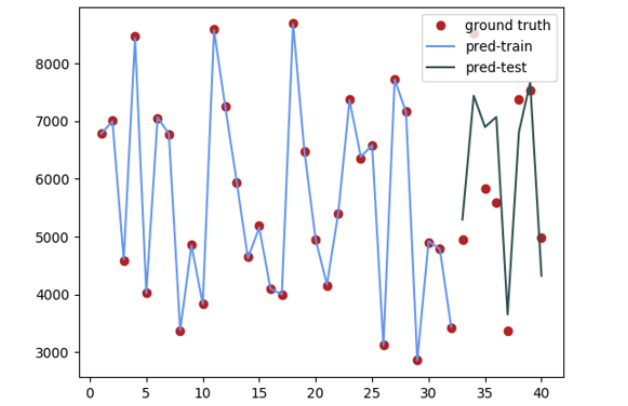

Figure 11. Resnet50 Prediction on Bayesian samples for power consumption

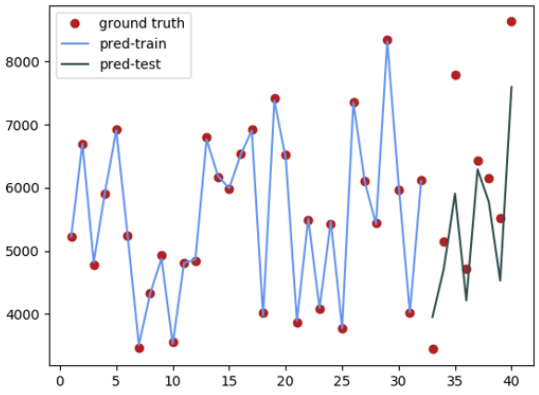

Figure 12. Resnet50 prediction on random samples for power consumption

That was expected given Bayesian Optimization relies heavily on Gaussian Process for exploration different regions with high variance. Similarly, figure 11 and figure 12 shows the comparison in predicting power consumption. The RMSE for power consumption predictions were as following: ***826.34 RMSE*** for samples obtained by Bayesian Optimization, and ***897.15 RMSE*** for random sampling. Notably, given the scale of power consumption values, the gap between RMSEs was not big enough. However, samples from Bayesian Optimization achieved more accurate prediction.

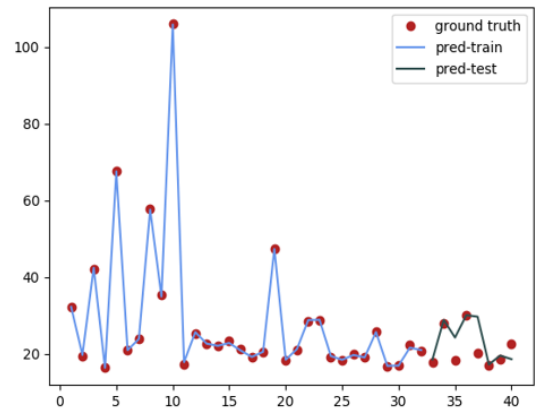

Figure 13. VGG16 prediction on Bayesian samples for inference Time

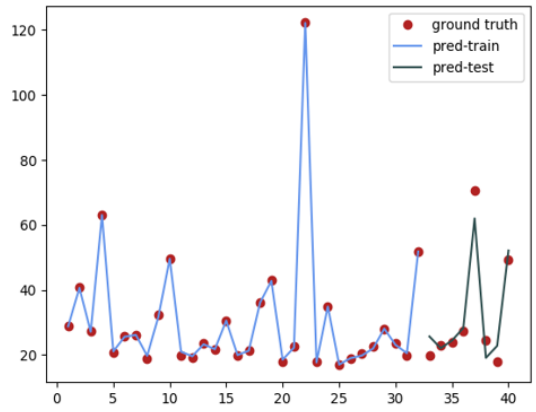

Figure 14. VGG16 prediction on random samples for inference Time

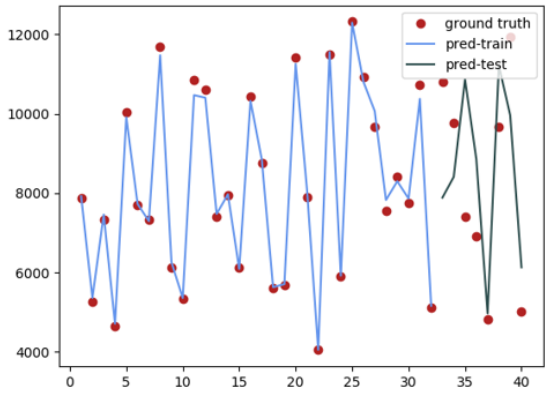

Figure 16. VGG16 prediction on random samples for power consumption

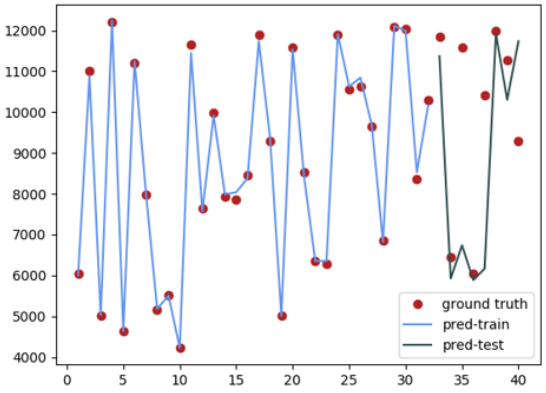

Figure 15. VGG16 prediction on Bayesian samples for power consumption

Likewise, on above figures, figure 13 and figure 14, are a comparison between Bayesian Optimization and Random sampling for predicting inference time taken from ***VGG16***. We calculated the RMSE for inference time predictions, and the outcomes as following: ***4.2 RMSE*** for samples obtained by Bayesian Optimization, and ***4.5 RMSE*** for random sampling. Clearly, Bayesian Optimization acheived more accuracy then random, but remarkably, the difference is negligible. On other hand, figure 15 and figure 16 shows the comparison in predicting power consumption. The RMSE for power consumption predictions were as following: ***2470.6 RMSE*** for samples obtained by Bayesian Optimization, and ***2052.4 RMSE*** for random sampling. That is in high contrast to the result obtained on ***Resnet50 model***. Explaining that, it could be the way Random design samples different configurations which resulted in various outcomes of power consumption. Additionally, while Random sampling achieved higher accuracy for power consumption for VGG16 model, it didn't achieved similarly for inference time.

6. Conclusion

We have presented different methods, both Beysian Optimization and Random Sampling, to explore the configuration space for multiple DNN models deployed in Nvidia Jetson TX1. Random Search and Bayesian Optimization are used to guide the search to optimize both inference time and power consumption.

We, also, utilized polynomial regression to understand the relationship between configurations parameters in regard to inference time and power consumption. To further solidify our experiments' result, we dug deep and compared the samples from Bayesian Optimization and Random design for accuracy using Gradient Boosting Regression. From the limited number of experiments we can observe that samples received from Bayesian Optimization on model Resnet50 trains better prediction model for both inference time and power consumption. However, we noticed that it has roughly lower RMSE for inference time and higher RMSE for power consumption.

In future works, we would like to try various Convolutional Neural Networks and Re-currant Neural networks with different workloads to further enhance our empirical study. Additionally, we hope to explore more configuration space in hardware-level such as CPU modes and tailgates, and one

compiler-level such CUDA.

7. Supplementary Material

The following bellow are the supplementary materials:

- 100 images from Cifar10 dataset
- Python code
- iPython Notebook
- Sampling data

References

Angelis, Lefteris, and George Stamatellos. "Multiple objective optimization of sampling designs for forest inventories using random search algorithms", 2004.

Brochu, Eric, Vlad M. Cora, and Nando De Freitas. "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning", 2010.

Swersky, Kevin, Jasper Snoek, and Ryan Prescott Adams. "Freeze-thaw Bayesian optimization", 2014.

Knudde, Nicolas and van der Herten, Joachim and Dhaene, Tom and Couckuyt, Ivo. "GPflowOpt: A Bayesian Optimization Library using TensorFlow", 2017.