CS6220 - Group 18 Project Proposal

Deep Neural Network Performance Optimization using GridSearch and GTDLBench

Group members: Junyan Mao, Jintong Jiang, Yiqiong Xiao, Gabriel Wang, Andrew Wang

October 1st, 2021

Motivation and Objectives:

In machine learning (ML), there are generally two types of parameters: model parameter (MP), and Hyperparameter (HP). MP is a set of configuration variables that are internal to the model, whose values can be estimated or learned from the data. HP refers to parameters that are used to control the learning process. HP optimization/tuning is a popular problem in ML, and it describes the process of picking a set of optimal HPs for an ML algorithm [*2*].

The issue with tuning HP is that we have no way to know the best value for it on a given problem before training. We must either use rules of thumb or infer value from similar previous problems. Without automatic HP tuning, practitioners and researchers need to make manual adjustments to the HPs over the course of multiple training runs to arrive at the optimal values, which would take a large amount of time for bigger datasets and more complex models [*3*].

There are currently a few industry-standard search algorithms that we can use to tune HPs: Random search (RS) – where we provide a statistical distribution for each HP, from which values may be randomly sampled; Grid search (GS) – where we specify a range of HP values and build a model for every possible HP combination, evaluate the model, then return the best set of HPs [*3*]. There are also many other optimization methods, such as evolutionary algorithms and Bayesian optimization. In this project, we aim to not only apply GS in HP tuning, but also other optimization techniques and use state-of-the-art tuning frameworks like LRBench[*6*], which is a tool for recommending a learning rate, and GTDLBench[*4*], which is a benchmarking system to measure the performance of deep learning frameworks. We also want to provide an interactive experience for users so that they will be able to adjust their HP configuration with ease.

Related Work:

There have been numerous studies conducted in deep learning due to the fast development and popularity in this field. Looking at previous studies, there exist frameworks such as GTDLBench[*4*] and LRBench[*6*]. The GTDLbench focuses on measuring the performance of different deep learning frameworks in terms of training time, testing time and accuracies. The LRBench specifically functions as a benchmarking recommendation tool that helps people

with selecting the LR policies in order to provide a more robust outcome. These provide features selection and help us with selecting the most optimal model.

Not only that, but there have also been multiple surveys and experiments run on the various optimization techniques for hyperparameter tuning that we may want to leverage. EMADE [*7*] was originally an evolutionary based AutoML framework that tunes the hyperparameters of shallow models and performs automatic data preprocessing. A later study expanded upon the EMADE framework specifically for deep neural network optimization [*5*], which again tuned hyperparameters and data preprocessing, but also neural architecture search. However, these frameworks utilized a tree representation of the networks, which may be limited compared to the traditional graph representations.

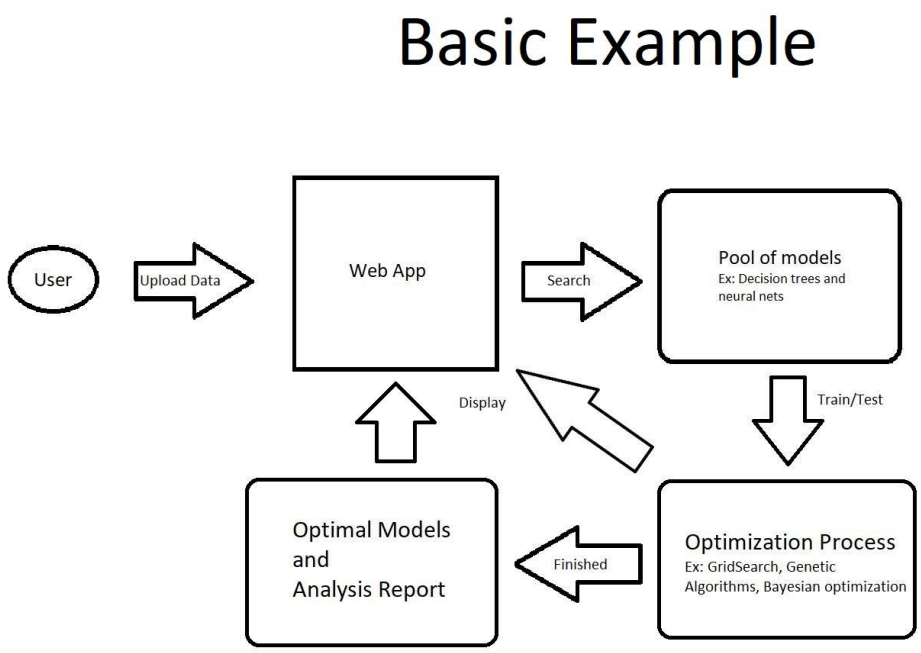
Besides the current tools that we can embed in our project, there also exists some deep learning workbench that shares similarities with what we envision our project would look like. OpenVINO[*1*] is a deep learning workbench that includes an interactive user interface that also supports a wide range of modern deep learning models. However, these software or web applications often don't focus too much on other functionalities such as providing analysis and allowing users to freely interact with the application. This functionality will be addressed in our project as well as implementing some novel techniques to tune and optimize the hyperparameters.

Proposed Work:

We aim to create an interactive web app and an AutoML framework. We want our product to have a highly interactive user experience, where users can upload any image dataset to our app and our app then searches for the optimal models to use and optimizes the hyperparameters of the models it finds. The web app will display the optimization process and the outputs of the models at various steps. The AutoML framework that we will create will automatically optimize shallow and deep learning models' hyperparameters using grid search, evolutionary algorithms, or Bayesian optimization. Shallow machine learning models will likely be implemented using the sklearn package, and deep neural networks will likely be implemented using pytorch and use successful architectures and pretrained weights such as AlexNet[*8*] or ResNet[*9*]. Once the optimization process is completed, the web app will return the optimal models and a brief analysis of the models.

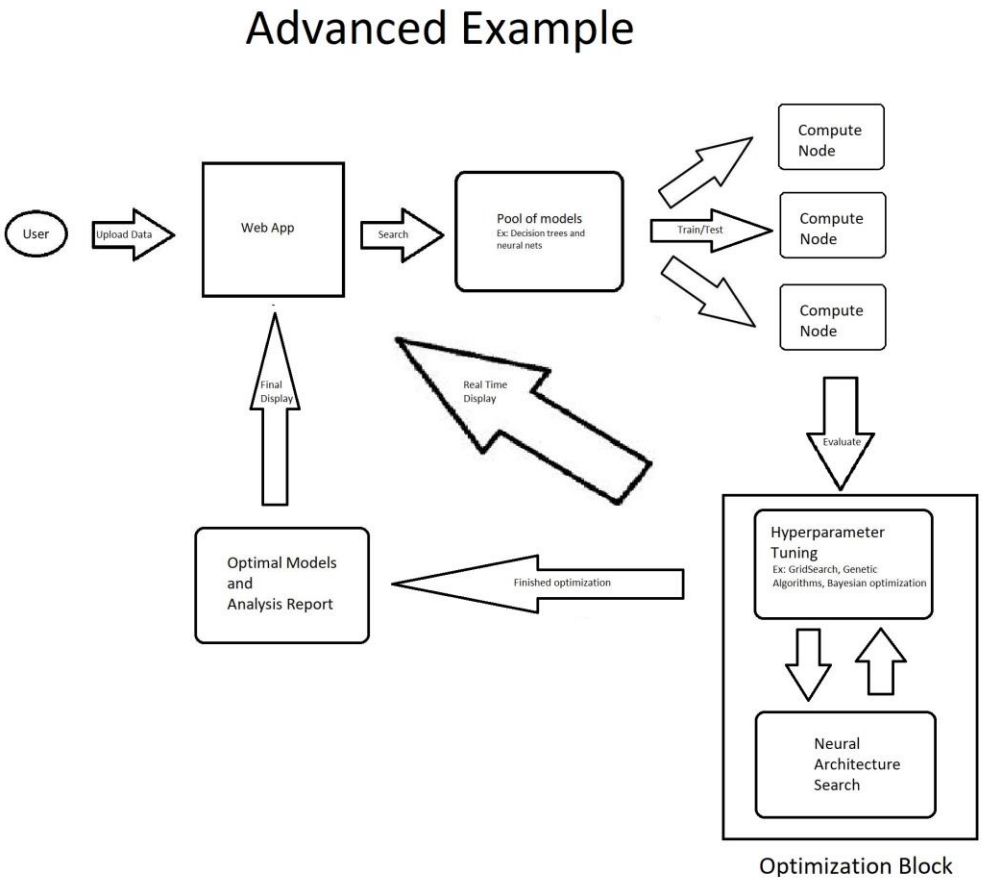
The users should be able to upload data and use the framework automatically, but also have customizable options to help guide the search and optimization process. For example, we could give the user options to select the type of optimization to use, seed the optimization process with the types of models to use, find and extract features from the data, or have options to stop the optimization process once a user set threshold is reached. However, to prevent the task from becoming too complex, we are limiting the user to just image datasets

for image classification purposes. The data should be properly cleaned and labeled before it is entered into the web app.



Success will be defined at various levels. The most basic level is creating the web app so that it runs locally and can take the user data, search and optimize models using grid search, and show optimization process and model outputs. The next level of success would be implementing evolutionary algorithms and Bayesian optimization for hyperparameter tuning and creating an automatic analysis report after the optimization is complete. Once these features have been successfully implemented, we can compare with benchmarks on well-known datasets and metrics such as GTDLBench and LRBench. If we successfully beat any state-of-the-art optimization processes or models in accuracy, training time, or testing time, then we can consider ourselves very successful!

We also have some stretch goals if we want to further improve our project past this point. For deep neural networks, we can utilize neural architecture search to further optimize the models and search for novel architectures. Should we utilize evolutionary algorithms for neural architecture search, there are plenty of modularity techniques that we can use to help the optimization process, and we have some novel ideas which could be applied to this framework. The deployed web app could also transfer data and models to a cluster or cloud resources for training and testing. This can allow our framework and app to be more scalable, to handle extremely large datasets or a large population of models to train.



The novelty of the proposed project arises from using new techniques to optimize the hyperparameters of the models, and the highly interactive user interface which also creates the analysis report. Grid search is a common technique for optimizing hyperparameters outside of hand-tuning. However, we aim to run an experiment to find if evolutionary algorithms or Bayesian optimization can better optimize the models. Should we accomplish our stretch goals, we believe that our neural architecture search method utilizing evolutionary algorithms with our individual representation and modularity techniques will be the first of its kind and still be scalable to handle larger datasets, assuming we can accomplish what we are envisioning.

Plan of Action:

Resources

- Operation System: Linux/MacOS
- Programming Languages: Python (scikit-learn, Pytorch, streamlit, flask)
- Hardware: Google Colab (model training), Cloud (AWS/Azure/GCP), Github/Heroku hosting

Date	Task	Deliverables
Week 1 (Sept 27 – Oct 3)	Finalizing project proposal, reviewing related works	Project Proposal
Week 2 (Oct 4 - Oct 10)	Meet with professor over proposal, refine proposal (if necessary), begin looking for datasets & setting up environment/workspace	
Week 3 (Oct 11 - Oct 17)	Dataset cleaning and starter web app frontend & backend	Dataset cleaning implementation and starter app
Week 4 (Oct 18 - Oct 24)	Implement shallow and deep models	Models
Week 5 (Oct 25 – Oct 31)	Optimization block (tune hyperparameters)	
Week 6 (Nov 1 – Nov 7)	Integrate ML model backend with frontend display	Functional local app
Week 7 (Nov 8 – Nov 14)	Testing and debugging	
Week 8 (Nov 15- Nov 21)	Finalize project & compare with benchmarks; If time, utilize cloud computing to optimize framework.	Complete project on Github
Week 9 (Nov 22 – 28)	Summarize methodology, documentation and work on presentation.	Presentation slides
Week 10 (Nov 29 – Dec 5)	Preparing for presentation	Project Presentation

Evaluation and Testing Method:

We will divide the testing and evaluation stage into Backend part and Frontend part. For backend, we will make sure that the shallow machine learning models, such as SVM, Decision Tree and Gaussian Naïve bayes, and the deep learning models, such as CNN and RNN, work properly with the default dataset. Also, we will test whether the grid search, evolution algorithms, and Bayesian optimization we implement actually tune the hyperparameters so that the models have better performance. We should make sure that the model running on the backend would update its running status and optimization process to the front end periodically. In the meanwhile, we would also test that the backend can generate automatic analysis reports after the optimization process is completed. For frontend, we will test whether users can upload the dataset/image successfully and pass them to our model. We would also test the ways of cleaning the data provided by the user. Also, the frontend should present the running status retrieved from the backend successfully. In the meanwhile, we will test whether it can

get the report generated from the backend and show this report on the main page so that user can download it or view it.

Bibliography:

1. Demidovskij, A., Gorbachev, Y., Fedorov, M., Slavutin, I., Tugarev, A., Fatekhov, M., & Tarkan, Y. (2019). OpenVINO Deep Learning Workbench: Comprehensive analysis and tuning of neural networks inference. *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. https://doi.org/10.1109/iccvw.2019.00104
2. Hyperparameter optimization. (2021). Retrieved from https://en.wikipedia.org/wiki/Hyperparameter_optimization
3. Hyperparameter tuning overview. (2021). Retrieved from https://cloud.google.com/ai-platform/training/docs/hyperparameter-tuning-overview
4. L. Liu, Y. Wu, W. Wei, W. Cao, S. Sahin and Q. Zhang, "Benchmarking Deep Learning Frameworks: Design Considerations, Metrics and Beyond," 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), 2018, pp. 1258-1269, doi: 10.1109/ICDCS.2018.00125.
5. Thite, A., Dodda, M., Liu, A., Agarwal, P., & Zutty, J. (2021). Concurrent neural tree and data preprocessing AutoML for image classification. Proceedings of the Genetic and Evolutionary Computation Conference Companion, 1990–1993. https://doi.org/10.1145/3449726.3463221
6. Wu, Y., Liu, L., Bae, J., Chow, K.-H., Iyengar, A., Pu, C., Wei, W., Yu, L., & Zhang, Q. (2019). Demystifying learning rate policies for high accuracy training of Deep Neural Networks. *2019 IEEE International Conference on Big Data (Big Data)*. https://doi.org/10.1109/bigdata47090.2019.9006104
7. Zutty, Jason. (2018). Automated machine learning: A biologically inspired approach.
8. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25. https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html
9. He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. ArXiv:1512.03385 [Cs]. http://arxiv.org/abs/1512.03385