Hyperparameter Optimization of Graph Neural Networks for the OpenCatalyst Dataset: A Case Study

Carmelo Gonzales *
Intel Labs

Eric Lee
SigOpt: An Intel Company

Kin Long Kelvin Lee
Intel Labs

Joyce Tang
SigOpt: An Intel Company

Santiago Miret
Intel Labs

Abstract

The proliferation of deep learning (DL) techniques in recent years has resulted in the creation of progressively larger datasets and deep learning architectures. As the expressive power of DL models has grown, so has the compute capacity needed to effectively train the models. One such example is the OpenCatalyst dataset in the emerging field of scientific machine learning, which has elevated the compute requirements needed to effectively train graph neural networks (GNNs) on complex scientific data. The extensive compute complexity involved in training GNNs on the OpenCatalyst dataset makes it very costly to perform hyperparameter optimization (HPO) using traditional methods, such as grid search or even Bayesian optimization-based approaches. Given this challenge, we propose a novel methodology for effective, cost-aware HPO. By leveraging a multi-fidelity approach on experiments with reduced datasets, we consider both hyperparameter importance and computational budget to show speedups of over 50 percent when performing the hyperparameter optimization of the E(n)-GNN model on OpenCatalyst.

1 Introduction

Over the last decade, machine learning has demonstrated the ability to significantly accelerate scientific discovery, especially through surrogate models that mitigate the cost of complex physical simulations. An exemplary case of this is speeding up quantum chemical calculations—*ab initio* using density functional theory (DFT)—that are required for AI-enabled materials design. Model architectural novelty, complexity, and scale have grown ambitiously, as evidenced by the OpenCatalyst dataset [Chanussot* et al., 2021] which contains terabytes of DFT calculations of catalytic materials. This project in particular has enabled the application of deep learning (DL) models for solid-state materials design and has inspired the creation of novel DL architectures for the field of geometric deep learning. Equivariant graph neural networks (GNNs), such as DimeNet [Klicpera et al., 2020], GemNet [Gasteiger et al., 2021], SE3-Transformer [Fuchs et al., 2020] and E(n)-GNN [Satorras et al., 2021] all strongly leverage physical inductive biases. The resulting geometric DL models are theoretically more expressive and thus easier to train with less data. However, these models require a significant amount of hyperparameter optimization (HPO) to obtain best performing configurations Snoek et al. [2012]. This challenge remains pervasive amongst DL tasks, especially in large computational scale tasks such as the OpenCatalyst dataset, whose models use a hyperparameter grid search to identify good hyperparameters Tran et al. [2022]. While automatic HPO tools—such as SigOpt, Optuna, and Weights and Biases—promise to increase the speed of HPO while abstracting its complexity away from users, naively using them to perform HPO on large, complex models may

*Correspondence to: <carmelo.gonzales@intel.com>

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

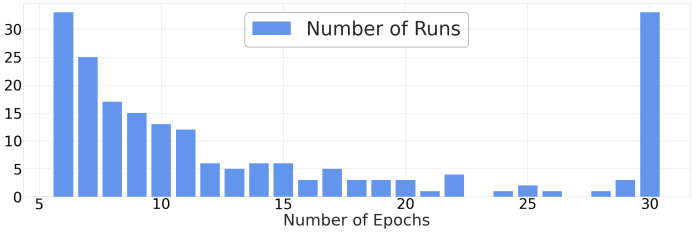

Figure 1: A histogram of the number of training epochs with early stopping enabled, where the max training epochs is thirty. We see that early stopping kicks in about 2/3 of the time and a time savings of about 50 percent on average.

result in an experiment that takes multiple months to complete. This is unfeasible for large-scale training runs, and this article details the procedures involved in completing the HPO of large scale GNNs trained on the OpenCatalyst dataset in a more reasonable (order of days) amount of time.

Our paper proposes a hyperparameter optimization method for training models on large-scale materials datasets, specifically the OpenCatalyst dataset, which are directly relevant to AI-Guided materials design. As deep learning becomes more and more sophisticated and materials related datasets become larger and larger in size, it will be increasingly important to leverage effective hyperparameter optimization methods to effectively perform model training. This will be particularly relevant when applying deep learning models in materials design frameworks that rely on deep learning based predictions, where accuracy of the prediction becomes significantly more important.

To perform HPO of large-scale scientific tasks, we modulate the size of our optimization experiments to generate cheaper data sources. We leverage these sources in multiple ways to inform the design of larger scale experiments and reduce the time needed for HPO by over fifty percent. The key takeaway here is no one method worked best "out-of-the-box" - only through a careful analysis of the model and data were we able to achieve these results. In Section 2 we discuss the problem setup, in Section 3 we breakdown the methodology used to solve this HPO problem, and finally, in Section 4 we discuss key results.

2 Problem Setup

The OpenCatalyst Project, jointly developed by Fundamental AI Research (FAIR) at Meta AI and Carnegie Mellon University's (CMU) Department of Chemical Engineering, encompasses one of the first large-scale datasets to enable the application of machine learning (ML) techniques, containing over 1.3 million molecular relaxations of 82 adsorbates on 55 different catalytic surfaces. The original release from 2019 has also been supplemented by subsequent updates in 2020 and 2022 and maintains an active leaderboard and annual competition [Chanussot* et al., 2021].

The challenge and dataset comprises three tasks. Here, we focus on the *S2EF* task, whereby an abstract DL model uses the initial, unrelaxed structure to predict both the electronic energy of the system and its first derivative with respect to atom positions (forces). We perform training experiments using the Open MatSci ML Toolkit framework [Miret et al., 2022] and the OpenCatalyst data splits, building from a 200K Training & 1M Validation split for the lowest fidelity, towards a 20M Training & 1M Validation dataset for the large scale experiments. For this task we utilize the E(n)-GNN architecture [Satorras et al., 2021]. E(n)-GNN is equivariant with respect to positions, which is useful for the S2EF task, where the rotation of the entire compound does not affect its properties.

3 Hyperparameter Optimization

The *hyperparameters* of a network encode the architecture and design of a neural network. Hyperparameters impact performance of a model—often in significant and opaque ways—and identifying performant hyperparameters is a crucial task during the model training process [Snoek et al., 2012].

In this section, we first outline some standard methods used to perform HPO. We then describe the ensemble of HPO methods used to effectively reduce HPO time by over fifty percent.

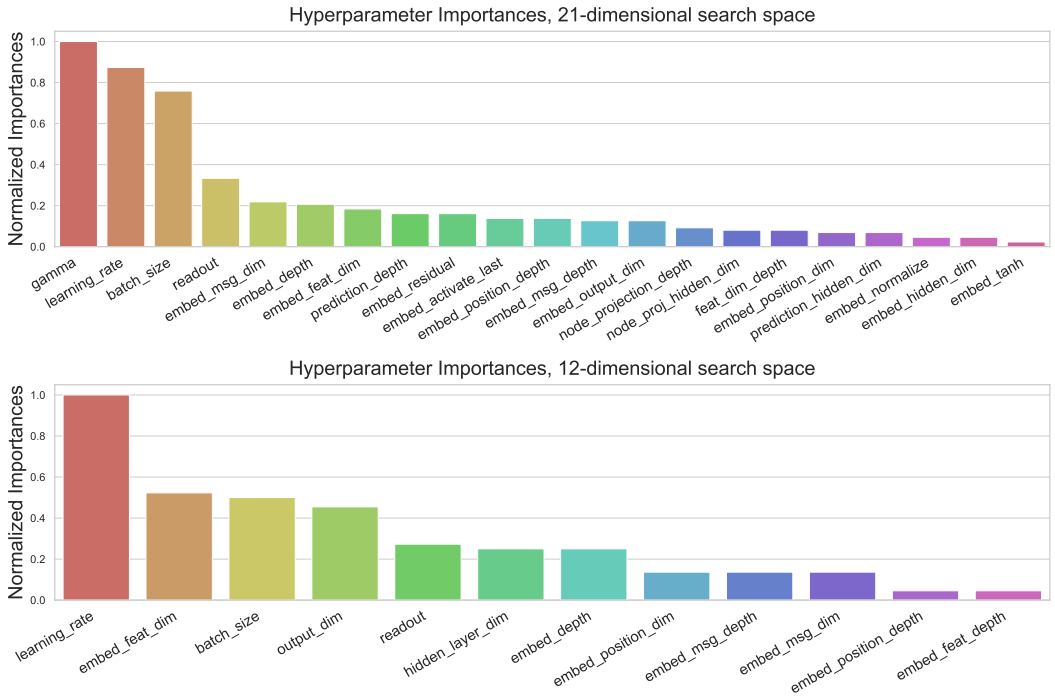

Figure 2: Above, we plot the importances of a 21-dimensional HPO search space (top) and a restricted 12-dimensional search space (bottom) of the E(n)-GNN Satorras et al. [2021]. The hyperparameters associated with training such as learning rate, gamma, or batch size are the most important. Furthermore, the layer depth hyperparameters seem to matter very little, suggesting that they ought to be fixed instead of selected for during the optimization process.

3.1 HPO Methods

Early Stopping: Early stopping is one of the standard mechanisms for saving wall clock time during training. Essentially, if the loss does not decrease after a fixed number of iterations (five in our case), training is terminated early. Note that early stopping incurs additional overhead because we must compute the validation metric every few epochs. However this is typically a very low overhead. In Figure 1, we found early stopping to decrease the total training time by about 50 percent on average (not included in the time savings provided by our method).

Bayesian Optimization: Hyperparameter optimization (HPO) frames the identification of performant model hyperparameters as the optimization problem $\min_{\theta \in \Theta} f(\theta)$, where Θ is the space of all possible hyperparameter configurations to search over and each θ is one particular hyperparameter configuration for which the optimization metric $f(\theta)$ may be computed post-training. Bayesian optimization (BO) Snoek et al. [2012] has been recognized as one of the most reliable and efficient ways of performing general HPO of an arbitrary machine learning model Turner et al. [2021].

Though BO is more efficient than other HPO methods, the problem of continuously training and retraining a large neural network for a hundred or more iterations is still a very large overhead, and makes it unsuitable for direct use in our situation.

Multi-fidelity Optimization: Multi-fidelity optimization [Forrester et al., 2007] introduces an additional *fidelity* parameter s which trades off noise and cost. A low-fidelity evaluation will be noisy but cheap, whereas a high-fidelity evaluation will be accurate but expensive.

In the context of this paper, fidelity parameters include the size of the training and validation sets or the number of training epochs. The question of how to best use low-fidelity information to accelerate optimization at higher fidelities is still a challenge at large in the optimization community. In general, the idea is to accrue a large amount of cheap data at the lower fidelity to accelerate optimization at the higher fidelity Li et al. [2017], Wu et al. [2020], Lee et al. [2020].

Hyperparameter Importance: Before proceeding with HPO at all, the tuneable hyperparameters must be selected. This decision controls the performance of the resulting model; there is a fundamental trade-off between the dimensionality of the search space and the expressiveness of models found in said search space. A high-dimensional search space is too large to search efficiently, and a low-dimensional one may not contain a suitable model. The question then becomes how to select a reasonable search space containing performant models that is not prohibitively large.

The analysis of variance (ANOVA), and its more efficient relative functional analysis of variance (FANOVA) Hutter et al. [2014], measures the importance of a parameter on a measured probabilistic outcome. In Figure 2, we show the FANOVA importances computed on different E(n)-GNN HPO experiments.

Computing these importances requires a nontrivial amount of data over the full search space of hyperparameter choices, but once we have these, we can rank hyperparameter importances and truncate the search space to search over only the top few contenders.

3.2 Our Methdology

We opt for an ensemble of the above methods to accelerate the hyperparameter optimization process. We fix two different fidelity levels: low and high (as opposed to a full continuum of fidelities). The low fidelity uses smaller training and validations. We perform the following steps:

1. Run Bayesian optimization on a large search space and at a low fidelity with early stopping.
2. Use this low-fidelity information to compute hyperparameter importances (FANOVA).
3. Save the top k (we set k to five) hyperparameters found at the low fidelity, and train the same k models at the high fidelity.
4. Pick the most important k hyperparameters, and perform hyperparameter optimization over those on the full fidelity.

The idea is to first identify promising candidate points in a large space quickly through low-fidelity optimization and early stopping, then to perform FANOVA testing to identify a smaller search space, and lastly perform high-fidelity optimization in this restricted search space with k promising candidates. The hope is that time savings achieved in the high-fidelity optimization will far exceed the overhead of obtaining the low-fidelity information. Certainly more extensive methods might accelerate the hyperparameter optimization further, but we found this ensemble to work well for our purposes.

4 Results & Discussion

4.1 Experimental Set Up

We consider the hyperparameter optimization and neural architecture search of the E(n)-GNN Satorras et al. [2021] models trained and validated on the OpenCatalyst dataset Chanussot* et al. [2021]. The low and high fidelities we consider are OpenCatalyst training sets of size 50k and 200k, respectively, with the same validation set. We performed all experiments using the Open MatSci ML Toolkit [Miret et al., 2022] on a cluster of GPUs.

We considered three experiments: a reference experiment in a large 21 dimensional space at medium fidelity which we use as a baseline, a low-fidelity, data-gathering experiment in a smaller 12 dimensional search space, and a high-fidelity experiment which uses our ensemble HPO methods. We refer to these as the "Reference-Baseline", "Low-Data", and "High-Method" experiments, respectively. See Appendix A.1 for the full search spaces of these experiments.

In Figure 1, we determined that the average number of epochs before early stopping kicked in was 18. We used this information by setting the number of training epochs to this number in the "High-Method" experiment, in the hopes of achieving a significant time savings without needing to early stop using a higher epoch count.

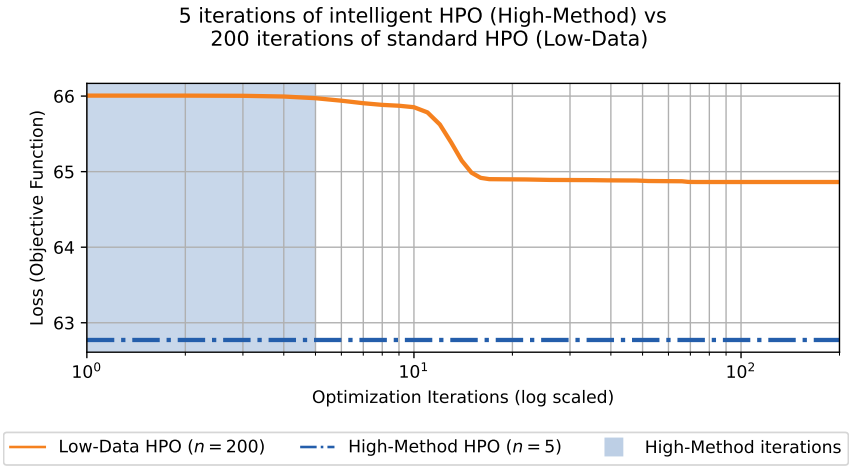

Figure 3: We compare 200 iterations of the Low-Data experiment (orange), which uses standard Bayesian optimization, with five iterations of the High-Method experiment (blue), which uses a more data-driven approach to HPO. We draw a horizontal line to indicate the best loss achieved by High-Method; observe that the Low-Data experiment never achieves a comparable loss even after 200 iterations, demonstrating the utility of our HPO method.

4.2 Results

We ran the Low-Data (low-fidelity) experiment as an information gathering step and then ran the High-Method (high-fidelity) experiment using the method described in Section 3.2. We then compared the results of this experiment with those in the Reference-Baseline experiment.

The Reference-Baseline experiment was able to achieve a best loss of about 65.5 in twelve iterations of HPO. In comparison, the High-Method experiment was able to achieve a best lost of 62.8 in only five iterations. We also need to measure the overhead of the Low-Data experiment, which ran 200 iterations of HPO on a training set of size 50k. We found training time to be roughly proportional to the size of the training set, and therefore the entire overhead of the Low-Data experiment was roughly equivalent to one iteration of the High-Method experiment. We conclude that our method was able to save more than fifty percent of real-world wall clock time in the optimization process. This does not include early stopping, which we estimate to further reduce cost by a factor of two. The real-world wall clock time saved on these experiments was significant: the High-Method experiments took two to three hours per iteration on average, while the Low-Data experiments took 61 minutes on average, see Tables 4, 5. This results in time savings on the order of days per experiment; had we used the training set with 20 million samples these savings may be even greater.

5 Concluding Thoughts

We have presented a case study of a simple hyperparameter optimization applied to the E(n)-GNN model for the OpenCatalyst Dataset task. We note that this case study is neither a comprehensive study of hyperparameter optimization (and neural architecture search) methods, nor a careful analysis of the challenges involving prediction of expensive DFT calculations. This case study instead is a bit of both, and though it lacks the depth of either the former or latter, we believe it provides valuable insight into the process of developing and optimizing for large-scale neural networks (without the need to make any specific claims or arguments).

The key takeaway from this case study is that a careful analysis of the dataset, the model, hyperparameter search space, and the optimization algorithm is required to identify a good model in a short amount of time —we believe no hyperparameter optimization method will generically work out-of-the-box without applying these principles.

References

Lowik Chanussot*, Abhishek Das*, Siddharth Goyal*, Thibaut Lavril*, Muhammed Shuaibi*, Morgane Riviere, Kevin Tran, Javier Heras-Domingo, Caleb Ho, Weihua Hu, Aini Palizhati, Anuroop Sriram, Brandon Wood, Junwoong Yoon, Devi Parikh, C. Lawrence Zitnick, and Zachary Ulissi. Open catalyst 2020 (oc20) dataset and community challenges. *ACS Catalysis*, 2021. doi: 10.1021/acscatal.0c04525.

Alexander IJ Forrester, András Sóbester, and Andy J Keane. Multi-fidelity optimization via surrogate modelling. *Proceedings of the royal society a: mathematical, physical and engineering sciences*, 463(2088):3251–3269, 2007.

Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in Neural Information Processing Systems*, 33:1970–1981, 2020.

Johannes Gasteiger, Florian Becker, and Stephan Günnemann. Gemnet: Universal directional graph neural networks for molecules. *Advances in Neural Information Processing Systems*, 34: 6790–6802, 2021.

Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *International conference on machine learning*, pages 754–762. PMLR, 2014.

Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. *arXiv preprint arXiv:2003.03123*, 2020.

Eric Hans Lee, Valerio Perrone, Cedric Archambeau, and Matthias Seeger. Cost-aware bayesian optimization. *arXiv preprint arXiv:2003.10870*, 2020.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

Santiago Miret, Kin Long Kelvin Lee, Carmelo Gonzales, Marcel Nassar, and Matthew Spellings. The open matsci ml toolkit: A flexible framework for machine learning in materials science. *arXiv preprint arXiv:2210.17484*, 2022.

Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pages 9323–9332. PMLR, 2021.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25*, pages 2960–2968, 2012.

Richard Tran, Janice Lan, Muhammed Shuaibi, Siddharth Goyal, Brandon M Wood, Abhishek Das, Javier Heras-Domingo, Adeesh Kolluru, Ammar Rizvi, Nima Shoghi, et al. The open catalyst 2022 (oc22) dataset and challenges for oxide electrocatalysis. *arXiv preprint arXiv:2206.08917*, 2022.

Ryan Turner, David Eriksson, Michael McCourt, Juha Kiili, Eero Laaksonen, Zhen Xu, and Isabelle Guyon. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In *NeurIPS 2020 Competition and Demonstration Track*, pages 3–26. PMLR, 2021.

Jian Wu, Saul Toscano-Palmerin, Peter I Frazier, and Andrew Gordon Wilson. Practical multi-fidelity bayesian optimization for hyperparameter tuning. In *Uncertainty in Artificial Intelligence*, pages 788–798. PMLR, 2020.

A Appendix

A.1 Experiment configurations

Tables in this section detail the three experiment configurations.

Table 1: Search space for Experiment *Reference-Baseline*; "grid" values in square brackets correspond to possible discrete values, while those separated by en dashes refer to minimum and maximum boundary values.

Parameter name	Values
Backbone parameters	
MLP hidden dim	[64, 128]
MLP output dim	[64, 128]
# of EGNN layers	2–5
Node MLP depth	2–3
Node MLP dim	[64, 128]
Edge MLP depth	2–3
Edge MLP dim	[64, 128]
Atom position MLP depth	2–3
Atom position MLP dim	[64, 128]
Residual term	[False, True]
Edge normalization	[False, True]
Edge tanh activation	[False, True]
Output activation	[False, True]
Graph read out	[Sum, Mean, Weighted sum, Max]
Node projection block depth	2–3
Node projection hidden dim	[64, 128]
Output block depth	2–3
Output hidden dim	[64, 128]
Optimizer parameters	
Adam LR	0.0001–0.01
Exponential LR scheduler decay	0.1–0.9
Batch size	8–16

Table 2: Search space for Experiment *Low-Data*; "grid" values in square brackets correspond to possible discrete values, while those separated by en dashes refer to minimum and maximum boundary values.

Parameter name	Values
Backbone parameters	
MLP hidden dim	[16, 32, 48, 64, 96, 128]
MLP output dim	[16, 32, 48, 64, 96, 128]
# of EGNN layers	2–5
Node MLP depth	2–4
Node MLP dim	[16, 32, 48, 64, 96, 128]
Edge MLP depth	2–4
Edge MLP dim	[16, 32, 48, 64, 96, 128]
Atom position MLP depth	2–4
Atom position MLP dim	[16, 32, 48, 64, 96, 128]
Output activation	[False, True]
Graph read out	[Sum, Mean, Weighted sum, Max]
Optimizer parameters	
Adam LR	0.0001–0.01
Batch size	[4, 8, 16, 24, 32]

Table 3: Search space for Experiment *High-Method*; "grid" values in square brackets correspond to possible discrete values, while those separated by en dashes refer to minimum and maximum boundary values. This search space was selected through by picking the seven most important hyperparameters as determined by FANOVA importance testing. We also restricted the search space for each of these parameters by checking previous experiments and determining a tighter bounding box for the top quantile of all hyperparameter configurations. Finally, we fixed all other hyperparameters to have value equal to the best performing configurations in the *Reference-Baseline* and *Low-Data* experiments.

Parameter name	Values
Backbone parameters	
MLP hidden dim	[16, 32, 48, 64]
MLP output dim	[16, 32, 48, 64, 96, 128]
Node MLP dim	[16, 32, 48, 64, 96, 128]
Edge MLP dim	[16, 32, 48, 64, 96, 128]
Optimizer parameters	
Adam LR	0.0001–0.006
Exponential LR scheduler decay	0.1–0.7
Batch size	[4, 8, 16, 24, 32]
Fixed parameters	
# of EGNN layers	3
Node MLP depth	2
Edge MLP depth	2
Edge normalization	0
Atom position MLP depth	64
Atom position MLP dim	2
Residual term	True
Edge tanh activation	False
Output Activation	True
Graph read out	Sum
Node projection block depth	2
Node projection hidden dim	128
Output block depth	3
Output hidden dim	64

A.2 Timings

In this section we provide some timing information associated with our experimentation. All timings are based off training with multi-GPU acceleration and the Miret et al. [2022] library.

Table 4: Below we list the average experiment time, average epoch time and their respective standard deviation across the 200 experiments in the Low-Data experiments. Note that early stopping is used in this experiment resulting in a high experiment time variance. Additionally, experiment time is also highly dependent on the hyperparameter configuration which explains the high epoch time variance. Each experiment was run on a single node containing 8 GPU's.

Epoch Time (m)	**Experiment Time (m)**
4.892 $\pm$3.177	61.367 $\pm$52.344

Table 5: Below we list the average validation time per batch, average validation time per epoch, average training time per batch, average training time per epoch, and total training time for five different hyperparameter configurations in the *High-Method* experiment. Units are given in parenthesis: either seconds (s), minutes (m), or hours (h). Each experiment was run using 6 nodes with 8 GPU's each. Note that the times greatly depend on the hyperparamter configuration; the purpose of this table is to simply to communicate the timings involved with experimentation.

Valid. Batch (s)	**Valid. Epoch** (m)	**Train Batch** (s)	**Train Epoch** (m)	**Total Train** (h)
0.018	0.765	0.153	7.168	2.380
0.016	1.903	0.091	8.970	3.262
0.016	1.713	0.099	9.242	3.286
0.016	1.648	0.112	9.779	3.428
0.017	1.975	0.161	12.078	4.216