Automated hyper-parameter optimization for deep learning framework to simulate boundary conditions for wave propagation

Harpreet Kaur, Sergey Fomel, and Nam Pham, The University of Texas at Austin*

SUMMARY

We propose a hyper-parameter optimization workflow for training the deep learning framework to simulate the effect of boundary conditions for wave propagation. Hyper-parameter selection is a crucial step in model building and has a direct impact on the performance of machine learning models. We implement three different hyper-parameter optimization techniques, namely random search, Hyperband, and Bayesian optimization, for the proposed network to simulate boundary conditions and compare the strengths and drawbacks of these techniques. The automated deep learning framework optimizes network training and significantly improves the efficiency of the workflow. The proposed method reduces human effort in the network tuning process, improves the performance of deep learning models by achieving the optimal minima, makes the model more reproducible, and can be extended to different deep learning based applications. Tests on different models verify the effectiveness of the proposed approach.

INTRODUCTION

Deep learning algorithms have demonstrated high performance in both research and commercial applications (Yu and Zhu, 2020). Different deep learning models are suitable for different types of problems and datasets. During the model tuning process, different model configuration arguments known as hyper-parameters are specified by the user to tune the network for a given problem statement. Hyper-parameters must be set prior to model training and can not be updated during the training process. Selecting the best hyper-parameters for models has a direct impact on the performance of the model (Yang and Shami, 2020). Conventionally, hyper-parameters are manually tuned by using an iterative trial and error based approach (Zöller and Huber, 2021). However, manual tuning is inefficient because of large number of hyper-parameters in complex models, nonlinear hyper-parameter interactions, and time-consuming model evaluations (Yang and Shami, 2020). These factors have inspired research on automated tuning of hyper-parameters known as hyper-parameter optimization (HPO).

HPO has multiple advantages over the manual hyper-parameter tuning process (Kaur et al., 2021). It reduces the considerable human effort in manual tuning of a large number of hyper-parameters, especially for complex models and large datasets. It improves the accuracy and efficiency of the models and makes the models and research more reproducible (Yang and Shami, 2020; Yu and Zhu, 2020). HPO problems need specialized optimization techniques because they can sometimes be non-convex and non-differentiable; therefore, traditional optimization techniques such as gradient descent may result in local instead of global minima (Luo, 2016). In addition, HPO algorithms should also be able to effectively handle different categories of hyper-parameters such as discrete (e.g., whether to use early stopping), continuous (e.g., learning rate), categorical (e.g., type of optimizer), etc.

Several studies proposed different categories of HPO methods such as decision theoretic methods, Bayesian optimization models, and multifidelity optimization techniques. These methods have their own advantages and limitations. Decision theoretic methods (Bergstra et al., 2011; Bergstra and Bengio, 2012) define a hyper-parameter search space and determine the best performing hyper-parameter combination by performing either an exhaustive search using grid search methods or a random selection using random search methods. These methods treat each parameter configuration independently and may require considerable time and space, especially the exhaustive grid search methods. Multifidelity optimization algorithms (Li et al., 2017) like Hyperband address the limited resource issue and in each iteration eliminates poorly performing hyper-parameter combinations to save computation time. However, similar to decision theoretic approaches, Hyperband also performs each evaluation independently of the previous evaluations and can thus perform unnecessary function evaluations in poorly performing regions. This issue can be mitigated by using Bayesian optimization (Eggensperger et al., 2013), which uses the history of previous evaluations to determine the next configuration. Bayesian models belong to the category of sequential models and are difficult to parallelize; however, they can usually converge to optimal hyper-parameter combinations within a few iterations.

In this work, we propose an HPO workflow for a deep learning framework to simulate boundary conditions for wave propagation. We perform a comparative analysis between three different HPO algorithms, namely random search methods, which are decision theoretic approaches, Hyperband optimization, which is a multifidelity optimization algorithm, and Bayesian optimization. We demonstrate that Bayesian optimization is more efficient and leads to a better performance in the testing phase than random search or Hyperband optimization. Although in this paper we demonstrate the HPO workflow on boundary condition simulation problem, the proposed workflow can easily be integrated into a variety of deep learning frameworks for different problem statements. Using numerical models of increasing complexity, we demonstrate that the proposed approach is efficient and converges to an optimal minima as opposed to the manual tuning of hyper-parameters.

HYPER-PARAMETER OPTIMIZATION

We implement the HPO workflow for a deep learning framework using UNet architecture (Ronneberger et al., 2015). We optimize two different categories of hyper-parameters, i.e., design hyper-parameters related to the construction of a deep learning model (number of filters, activation function, etc.) and

10.1190/image2022-3743090.1

optimizer hyper-parameters related to the training process of the model (learning rate, dropout rate, mini-batch size, number of epochs, etc.). For the design-based hyper-parameters, we provide several filters and activation functions, and the optimization process selects the best combination on the basis of the complexity of the problem. Next we specify the search space for following optimizer hyper-parameters:

Learning rate - It is one of the most important hyper-parameters in a deep learning model, which defines step size for the convergence of the iterative process. A high learning rate can accelerate the training process but may also oscillate the gradients and hinder convergence, whereas a low learning rate leads to smooth convergence with a longer training time. We provide a range of learning rates (Table 1) and the optimization algorithm selects the appropriate learning rate to converge the objective function to global minima.

Dropout rate - Dropout is a method of stochastic regularization, which prevents overfitting and enables the neural network to learn robust features (Srivastava et al., 2014). It is a computationally efficient method to simulate an ensemble of neural networks by randomly dropping neurons during the training process. We use a dropout range of 0.0 to 0.5 with a step size of 0.05, and the optimization algorithm finds the optimal dropout rate for the neural network training.

Batch size - Batch size is the number of training samples used to estimate the gradient. Selection of batch size impacts training stability and generalization performance (Bengio, 2012). Smaller batch size offers a regularization effect and lowers generalization error. We incorporate different batch sizes in the search space for the optimization algorithm, and the workflow automatically selects the optimal batch size for training.

Table 1: Hyper-parameters and their respective values.

Filters	Activation functions	Batch size	Dropout rate	Learning rate
32-64	Sigmoid ReLu, tanh	8-16	0-0.5	0.001-0.01

We initiate the optimization process using the earlier defined search space, and optimize the hyper-parameters using three optimization algorithms:

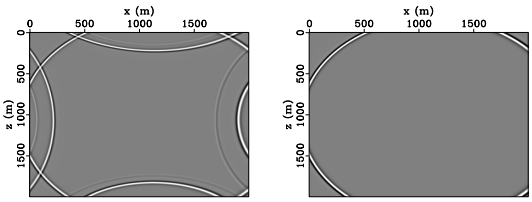
1) Decision theoretic approach-The first algorithm that we implement is a random search, which is a type of decision theoretic method. Random search algorithm randomly selects the hyper-parameter configurations in the search space defined by the user and trains these candidate values until the defined trials are exhausted. It can cover a large search space; however, it treats each evaluation independently and does not keep track of the previously well-performing regions. We implement random search with 20 trials, and discuss the test results of the best performing model in the next section.

2) Multifidelity optimization-We further improve the efficiency of the random search algorithm by using multifidelity optimization. One of the issues with random search method is that with a larger hyper-parameter configuration space, the longer execution time may not let the algorithm converge to global minima if the resources are limited. To overcome this limitation, Hyperband achieves a trade-off between the number of hyper-parameter configurations and their allocated budget by discarding the poorly performing configurations in each iteration using a successive halving method and passing only well performing configurations to the next iteration. We compare test results in the next section.

3) Bayesian optimization- Earlier mentioned algorithms independently process each hyper-parameter configuration and may thus end up wasting a lot of computational time in poorly performing regions of the search space. To circumvent this issue, we implement Bayesian optimization. Unlike earlier mentioned algorithms, Bayesian optimization determines the next hyper-parameter configuration on the basis of previously tested configurations and, thus, given appropriate resources, it assures convergence toward the optimal configuration (Yang and Shami, 2020). Bayesian optimization has two key components: a surrogate model that fits observed points into the objective function and an acquisition function that provides a trade-off between exploration and exploitation regions. We first obtain the predictive distribution for the probabilistic surrogate model. Next we use the acquisition function for exploration where we sample instances in regions that have not been previously sampled, as well as exploitation, where we sample the most promising regions with higher likelihood of global optima. The advantage of using Bayesian optimization is that it can detect the optimal hyper-parameter configuration on the basis of previously tested values. However, dependence on the previously tested values makes the model sequential and difficult to parallelize.

We train the network in two steps: hyper-parameter optimization and parameter tuning, where parameter tuning updates weights and biases. We create training labels using different isotropic and anisotropic models with different source wavelet frequencies using 2,500 time slices from 5 shot locations at the corner and in the center of the model. We train the network to learn the mapping between the bounded model consisting of spurious reflections and the padded model used to simulate the unbounded domain (Figures1). We first define hyper-parameters for the deep learning model, along with the search space. Next, we implement hyper-parameter optimization workflow using random search, Hyperband, and Bayesian optimization workflows with 20 trials. After optimizing the hyper-parameters, we pick the best hyper-parameter configuration from the leader-board of models (green line in Figure 2) and use it to tune the parameter for network training. We incorporate K-fold cross-validation in the training process to analyze the generalization ability of the proposed model. For this work, we choose $K = 5$, which as shown by Kuhn et al. (2013), has low bias and modest variance. We fit the model using $K-1$ folds and validate the data with the remaining Kth fold. We repeat this process until every Kth-fold serves as the test set, and we note down the error for each fold. The mean square of errors (MSE) from all iterations gives us the cross-validation (CV) performance metric given as $CV_k = \frac{1}{k}\sum_{i=1}^{k} MSE_i$. We obtain values of 0.0027, 0.0023, 0.0035, 0.0027, and 0.0029 for different folds with an average value of 0.0028, which indicates that we do not have a split bias and the network generalizes well on the unseen data. During training, the network learns to attenuate spurious boundary reflections. Then we test the trained network using the best hyper-parameter configura-

tion by using the three optimization algorithms and perform a comparative analysis during the testing phase in the following examples.



(a) (b)

Figure 1: One of the training slices for HPO and parameter tuning. The network learns mapping from wavefield in bounded domain (a) to the wavefield in unbounded domain simulated by padded model (b) using different HPO workflows. The trained network is then tested for comparative analysis between different HPO algorithms.

EXAMPLES

Gradient velocity model

We first use a synthetic model with large velocity variations (Figure 3a). The model is discretized on a 400 $\times$ 400 grid with a 5-m spacing along vertical and horizontal directions (Sun et al., 2016). We simulate the wavefield by using a low-rank extrapolation operator (Fomel et al., 2013) with a 20-Hz Ricker wavelet and a time step of 2 ms. A blind test slices is shown in Figure 3b, and the ground truth (not seen by the network) is shown in Figure 3c. We test the trained network to analyze output using the three HPO algorithms, namely, random search, Hyperband, and Bayesian optimization (Figures 3d, 3e, and 3f) with a constrained number of trials. The test output using the random search algorithm needs further improvement (marked by the red arrow in Figure 3d). Next, we analyze the test results using Hyperband algorithm (Figure 3e). Hyperband eliminates poorly performing hyper-parameter configurations in each run, and, therefore, in a given number of trials it can converge to near optimal minima, as compared to the random search method. However, we still have some remnant spurious reflections (red arrows in Figure 3e). Next, we test the Bayesian optimization model (Figure 3f). In the same number of trials as the other two methods, the Bayesian optimization model converges to an optimal minima, and the output is close to the ground truth (Figure 3c). To quantify the output using the three optimization methods, we compute the structural similarity index (SSIM) (Wang et al., 2004) and correlation coefficient (R^2) (Table 2) for the test cases, which indicates that the Bayesian optimization determines a more appropriate hyper-parameter configuration because it takes into account previous evaluations, and with each iteration it moves closer to the optimal minima.

Marmousi Model

In the previous example, we observed that Bayesian optimization produces the best hyper-parameter configuration, and its output is close to ground truth. We test the best picked model using Bayeisan optimization on the Marmousi model (Versteeg, 1994). The model is discretized on a 376 $\times$ 384 grid with a spatial sampling of 25-m along horizontal and vertical directions (Kaur et al., 2019, 2021). We simulate the wavefield using a low-rank extrapolation operator (Fomel et al., 2013) with a 15-Hz Ricker wavelet and a time step of 2 ms. The test wavefields with spurious reflections for two different shot locations are shown in Figures 4a and 4d. The output using the Bayesian optimization workflow (Figures 4b and 4e) is free of the boundary reflections and wrap-arounds and is close to the output using the unbounded media simulated by padded model (Figures 4c and 4f).

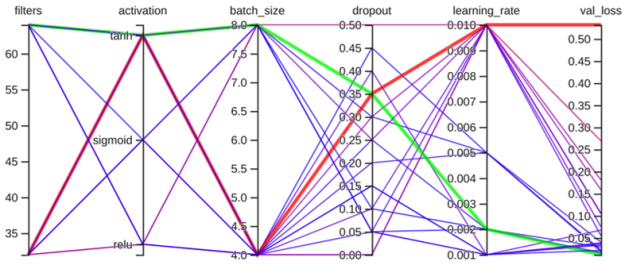


Figure 2: HPO with Bayesian optimization workflow. Optimization workflow considers dynamic interaction between hyper-parameters and converges to optimal minima. Green line shows best combination of hyper-parameters (with lowest loss) and red line shows worst performing hyper-parameter combination (with highest loss).

Table 2: SSIM and correlation coefficient for test case: Comparative analysis between three HPO algorithms.

Metrics	Random Search	Hyperband	Bayesian
SSIM	0.87	0.95	0.97
R^2	0.87	0.96	0.98

CONCLUSIONS

We have developed an automated hyper-parameter optimization workflow for a deep learning framework. We train the network using the best picked model from the hyper-parameter optimization workflow and perform a comparative analysis between three different optimization algorithms; random search, Hyperband, and Bayesian optimization. We demonstrate that for a given search space and computational resources, Bayesian optimization performs better than random search or Hyperband because for each hyper-parameter configuration, it takes into account the previous evaluation, which helps convergence within a few iterations. The proposed algorithm reduces manual input into training process, optimizes values of hyper-parameters for a deep learning framework, and thus maximizes the predictive accuracy of deep learning models. We show application of the proposed workflow for simulating the boundary condition for wave propagation; however, the proposed hyper-parameter optimization workflow can easily be integrated into a variety of deep neural networks for different problems.

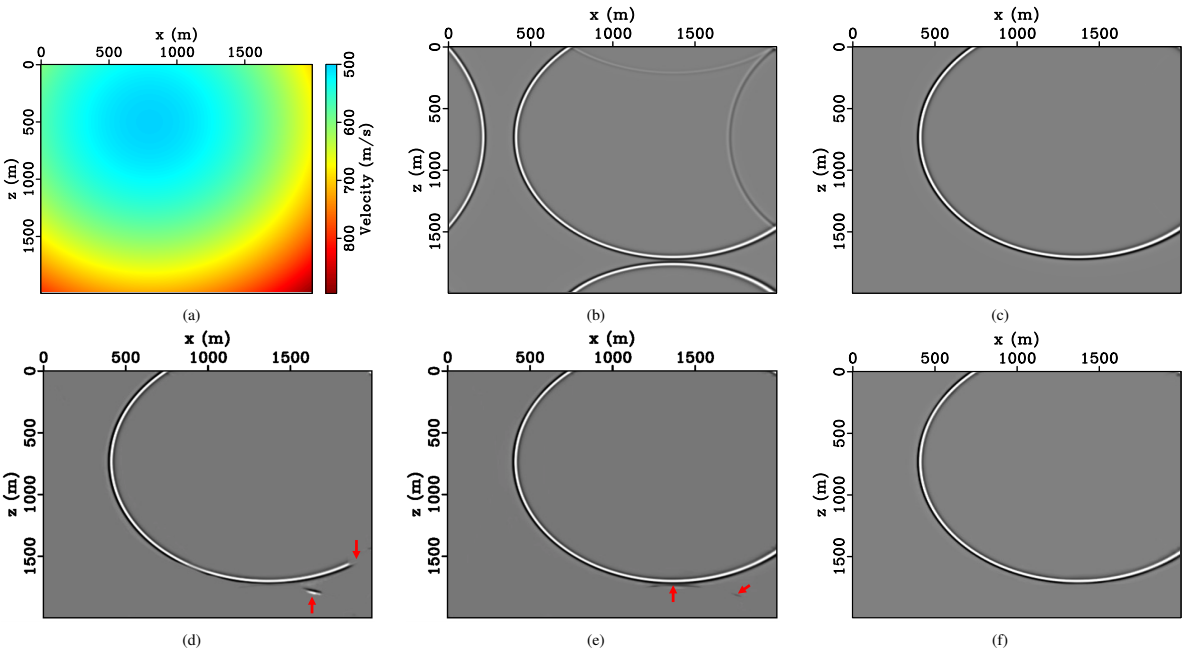


Figure 3: Blind-test results for gradient velocity model at t=1.82 s (shot locations and time slices are not part of training); (a) gradient velocity model, (b) wavefield time slice for bounded media with boundary reflections and wrap-arounds, (c) wavefield time slice for unbounded media simulated using padded model. Output wavefields free from boundary reflections and wrap-arounds using (d) random search optimization, (e) Hyperband optimization, and (f) Bayesian optimization. Red arrows indicate the regions where random search and Hyperband optimization needs further improvement.

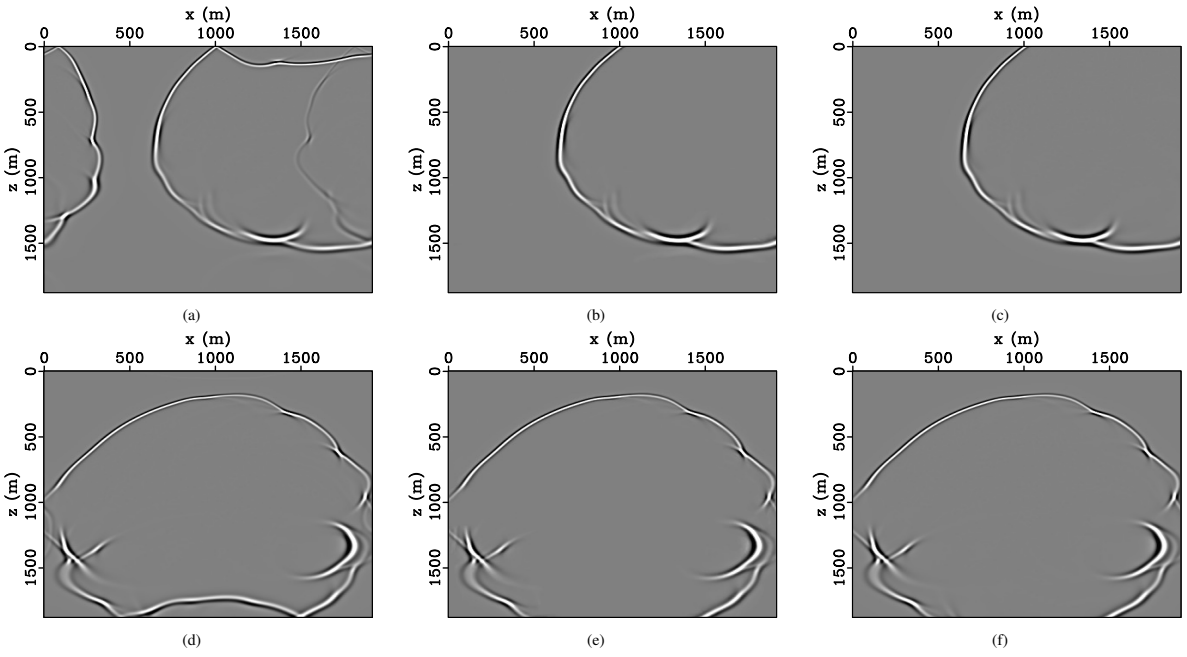


Figure 4: Marmousi model test data for network trained using Bayesian optimization. Wavefield snapshot at t=1.47 s: (a) wavefield snapshot with boundary reflections, (b) wavefield snapshot using proposed method, and (c) wavefield snapshot without boundary reflections in unbounded media simulated by padding the model. Wavefield snapshot at t=1.524 s for different blind-test shot location: (d) wavefield snapshot with boundary reflections, (e) wavefield snapshot using proposed method, and (f) wavefield snapshot without boundary reflections in unbounded media simulated by padding the model.

10.1190/image2022-3743090.1

References

Bengio, Y., 2012, Practical recommendations for gradient-based training of deep architectures: Springer.

Bergstra, J., R. Bardenet, Y. Bengio, and B. Kegl, 2011, Algorithms for hyper-parameter optimization: Advances in Neural Information Processing Systems, **24**, 10495258.

Bergstra, J., and Y. Bengio, 2012, Random search for hyper-parameter optimization: Journal of Machine Learning Research, **13**, 281–305.

Eggensperger, K., M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, and K. Leyton-Brown, 2013, Towards an empirical foundation for assessing Bayesian optimization of hyperparameters: Presented at the NIPS workshop on Bayesian Optimization in Theory and Practice.

Fomel, S., L. Ying, and X. Song, 2013, Seismic wave extrapolation using lowrank symbol approximation: Geophysical Prospecting, **61**, 526–536, doi: https://doi.org/10.1111/j.1365-2478.2012.01064.x.

Kaur, H., S. Fomel, and N. Pham, 2019, Overcoming numerical dispersion of finite-difference wave extrapolation using deep learning: 89th Annual International Meeting, SEG, Expanded Abstracts, 2318–2322, doi: https://doi.org/10.1190/segam2019-3207486.1.

Kaur, H., S. Fomel, and N. Pham, 2021, Boundary conditions for acoustic and elastic wave propagation using deep learning: First International Meeting for Applied Geoscience & Energy, SEG/AAPG, Expanded Abstracts, 1390–1394, doi: https://doi.org/10.1190/segam2021-3580865.1.

Kuhn, M., and K. Johnson, 2013, Applied predictive modeling: Springer.

Li, L., K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, 2017, Hyperband: A novel bandit-based approach to hyperparameter optimization: The Journal of Machine Learning Research, **18**, 6765–6816, doi: https://doi.org/10.48550/arXiv.1603.06560.

Luo, G., 2016, A review of automatic selection methods for machine learning algorithms and hyper-parameter values: Network Modeling Analysis in Health Informatics and Bioinformatics, **5**, 1–16, doi: https://doi.org/10.1007/s13721-016-0125-6.

Ronneberger, O., P. Fischer, and T. Brox, 2015, U-net: Convolutional networks for biomedical image segmentation: International Conference on Medical Image Computing and Computer-assisted Intervention, 234–241, doi: https://doi.org/10.48550/arXiv.1505.04597.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, 2014, Dropout: a simple way to prevent neural networks from overfitting: The Journal of Machine Learning Research, **15**, 1929–1958.

Sun, J., S. Fomel, and L. Ying, 2016, Low-rank one-step wave extrapolation for reverse time migration: Geophysics, **81**, S39–S54, doi: https://doi.org/10.1190/geo2015-0183.1.

Versteeg, R., 1994, The Marmousi experience: Velocity model determination on a synthetic complex data set: The Leading Edge, **13**, 927–936, doi: https://doi.org/10.1190/1.1437051.

Wang, Z., A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, 2004, Image quality assessment: from error visibility to structural similarity: IEEE Transactions on Image Processing, **13**, 600–612, doi: https://doi.org/10.1109/TIP.2003.819861.

Yang, L., and A. Shami, 2020, On hyperparameter optimization of machine learning algorithms: Theory and practice: Neurocomputing, **415**, 295–316, doi: https://doi.org/10.1016/j.neucom.2020.07.061.

Yu, T., and H. Zhu, 2020, Hyper-parameter optimization: A review of algorithms and applications: arXiv preprint arXiv:2003.05689.

Zöller, M.-A., and M. F. Huber, 2021, Benchmark and survey of automated machine learning frameworks: Journal of Artificial Intelligence Research, **70**, 409–472, doi: https://doi.org/10.1613/jair.1.11854.