

**NAME – Ishan Singh**

**ROLL NO. – 20CH10026**

**BTP – I**

**Mid-Sem Report**

**Prof. Jhareswar Maiti**

**DEPARTMENT OF**

**INDUSTRIAL AND SYSTEMS ENGINEERING**

**TABLE OF CONTENTS**

**1. Introduction**  3

1.1 Hyperparameter Optimization 3

1.2 Advantages of HPO 3

1.3 Applications of HPO 4

**2. Research Objective**  4

**3. Dataset**  5

**4. CNN Model Architecture** 6

**5. Random Search**  7

5.1 Optimized Hyperparameters 7

5.2 Plots 8

5.3 Results 9

5.4 Discussions 10

1. **Bayesian Optimization**  11

6.1GPEI Flowchart 12

6.2 Optimized Hyperparameters 13

6.3 Plots 14

6.4 Results 15

6.5 Discussions 16

**7. Multilevel Evolution Optimization**  17

7.1 MLEO Flowchart 19

7.2 Optimized Hyperparameters 20

7.3 Plots 21

7.4 Results 22

7.5 Discussions 23

**8. Combination of MSMLEO\_GPEI**  24

**9. Conclusions**  25

**10. References** 26

**INTRODUCTION**

**Hyperparameter Optimization**

Hyperparameter optimization is the process of selecting an ideal set of hyperparameters to achieve maximum performance on data in a reasonable amount of time. Hyperparameters are external configurations for the algorithm that are not learned from the data and which are used to control the learning process. They play an important role in the prediction accuracy of an algorithm. They might include learning rate, epoch number, batch size, and so on, in neural networks.

The various techniques employed in hyperparameter optimization are:

* **Grid Search:** It is a very simple method used for hyperparameter tuning, where the model is trained for all the possible combinations of hyperparameters and the set of hyperparameters that gives the best performance is selected.
* **Random Search:** It is an alternate technique to the exhaustive Grid Search, used for hyperparameter tuning where random combinations of hyperparameters are used to find the best solution for the built model.
* **Bayesian Optimization:** It is an optimization algorithm using an advanced probabilistic model designed for finding the minimum of a function that is expensive to evaluate. It is particularly useful for optimizing hyperparameters of machine learning models.
* **Gradient-Based Optimization:** It calculates gradients with respect to hyperparameters and optimizes them using gradient descent.
* **Evolutionary Algorithms:** They use mechanisms inspired by biological evolution, such as mutation, crossover, and selection, to optimize hyperparameters.

**Advantages of Hyperparameter Optimization**

1. **Enhanced Model Accuracy:**

* **Precision Tuning:** Hyperparameter optimization refines the model's parameters, enhancing its predictive accuracy and generalization capabilities on unseen data.
* **Customization:** Each dataset is unique, and through optimization, models can be finely tuned to accommodate the peculiarities of the data they are working on, resulting in superior performance.

2. **Efficient Resource Utilization:**

* **Time-Saving:** Automated hyperparameter optimization drastically cuts down the time traditionally spent on manual tuning, leading to quicker model deployment.
* **Compute Resources:** Intelligent search spaces and optimization strategies ensure that computational resources are used efficiently, reducing costs.

3. **Automation and Scalability:**

* **Hands-Off Approach:** Once set up, the optimization process is often autonomous, requiring minimal human intervention.
* **Scalable:** Can be easily scaled to handle optimization for complex models and large search spaces without significant manual labour.

**Applications of Hyperparameter Optimization**

1. **Deep Learning:**

* **Neural Networks Configuration:** Selecting appropriate activation functions, learning rates, and the number of layers and neurons in each layer in neural networks.
* **Convolutional Neural Networks (CNNs):** Very helpful in tuning of the filter sizes, strides, padding, and pooling layers to improve image classification and recognition tasks.

2. **Ensemble Learning and Tree-Based Models:**

* **Boosting Algorithms:** Used to optimize algorithms like XGBoost, AdaBoost and Gradient Boosting for tasks like regression, classification, and so on.
* **Random Forests:** Applicable in adjusting the number of trees, tree depth, and other parameters to enhance the model's predictive power and improvement in accuracy.

3. **Natural Language Processing (NLP):**

* **Text Classification:** Optimizing parameters for algorithms used in sentiment analysis, topic modelling, and document classification.
* **Sequence Models:** Tuning hyperparameters for recurrent neural networks (RNNs) and transformers to improve performance on sequence-to-sequence tasks.

**Research Objective**

Development of a hybrid optimization method combining Bayesian optimization and a multiscale and multilevel genetic algorithm to optimize the hyperparameters on multiscale grids for machine learning performance enhancement.

In our experiment, we are using the CIFAR-10 dataset to demonstrate the optimization efficiency of various algorithms, namely GPEI Bayesian Optimization (Gaussian Process Expected Improvement) and MLEO (Multilevel Evolution Optimization). These advanced optimization techniques are being used to compare against a base Convolutional Neural Network (CNN) model, which is devoid of any optimization enhancements. This comparative analysis aims to shed light on the performance improvements brought about by the incorporation of GPEI and MLEO algorithms.

**CIFAR-10 Dataset**

The CIFAR-10 dataset is a collection of images that are commonly used for machine learning and computer vision applications. The dataset was created by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

**Key Details:**

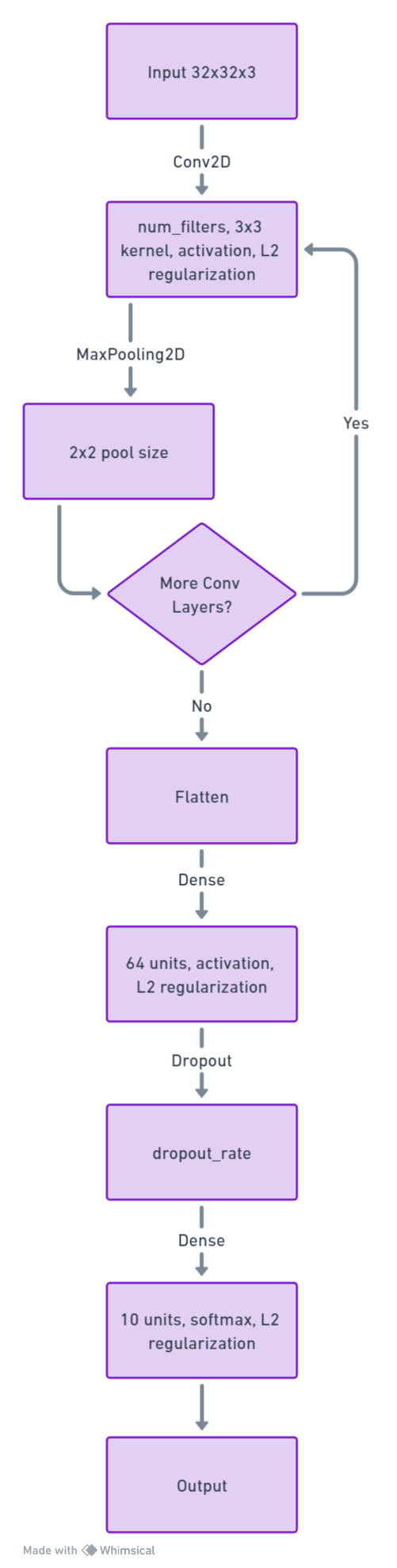
1. **Number of Images**: The CIFAR-10 dataset consists of 60,000 32x32 color images in 10 different classes.
2. **Classes**: The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images of each class.
3. **Training and Test Set**: The dataset is divided into a training set and a test set. The training set contains 50,000 images, while the test set contains 10,000 images.
4. **Data Format**: Images are stored in pixel level in the form of a 3-dimensional array, with the dimensions 32x32x3. The three channels represent the RGB values of the colors.
5. **Label Format**: Each image is labelled with a single integer that corresponds to the class of the object in the image.

**Usage:**

CIFAR-10 is widely used for training and testing machine learning models, including:

* **Classification**: Building models to correctly classify images into one of the 10 classes.
* **Feature Learning**: Learning meaningful features or representations from the images.
* **Transfer Learning**: Using pre-trained models on CIFAR-10 for other image classification tasks.
* **Data Augmentation Techniques**: Testing and developing new techniques for artificially expanding the dataset.

**Architecture of CNN Model used**

****

**Results of Hyperparameter Optimization by Random Search on Base CNN model**

|  |  |
| --- | --- |
| Hyperparameters | Optimized Value |
| Number of convolutional layers | 2 |
| Number of Filters | 32 |
| Dropout Rate | 0.6624 |
| Optimizer | adagrad |
| Learning rate | 0.0062 |
| Activation function | tanh |
| Epochs | 34 |
| Batch size | 32 |
| Kernel size | 3 x 3 |
| Regularization Lambda | 0.045 |

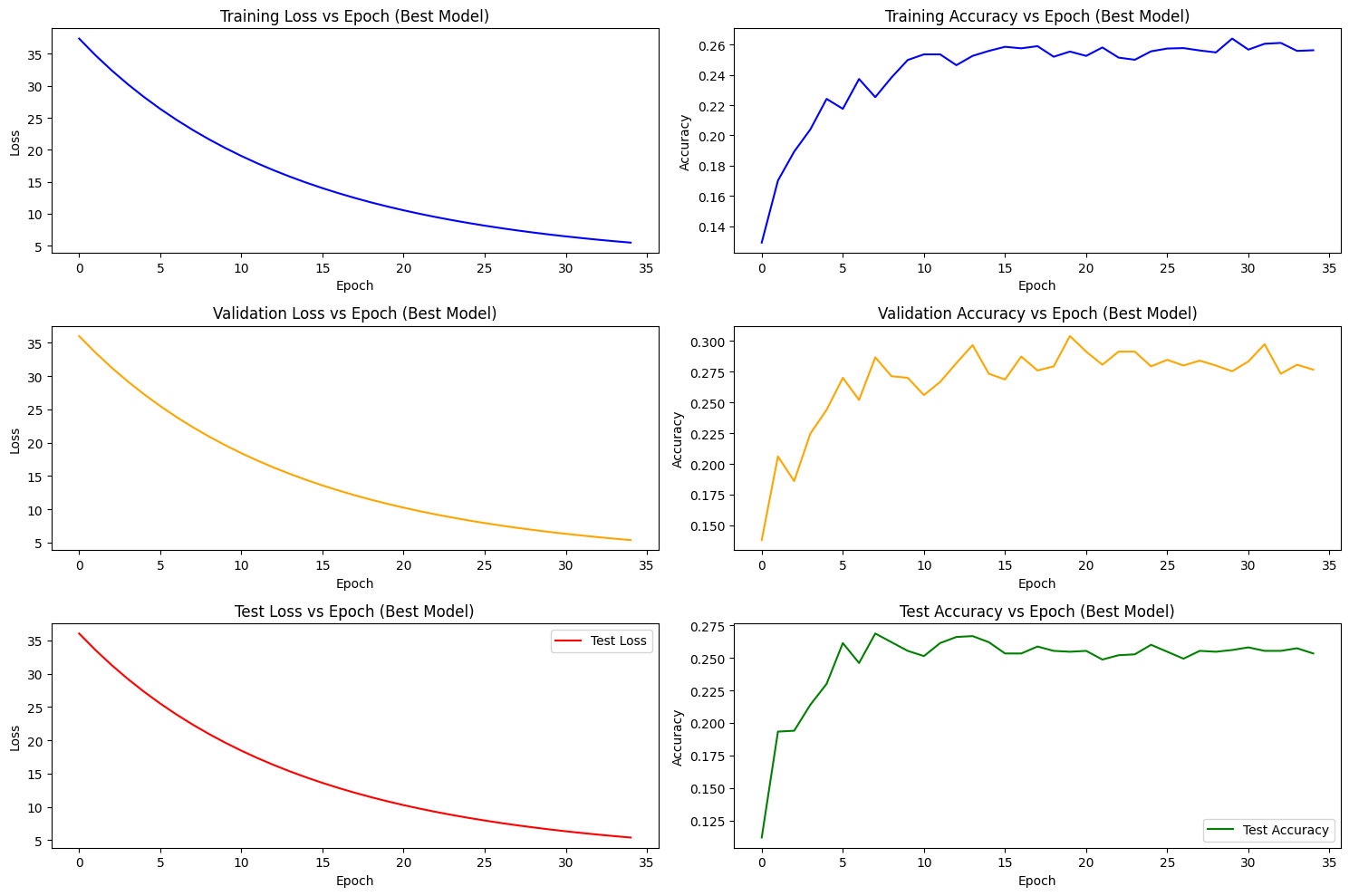
Maximum Training accuracy: 0.2634 = 26.34%

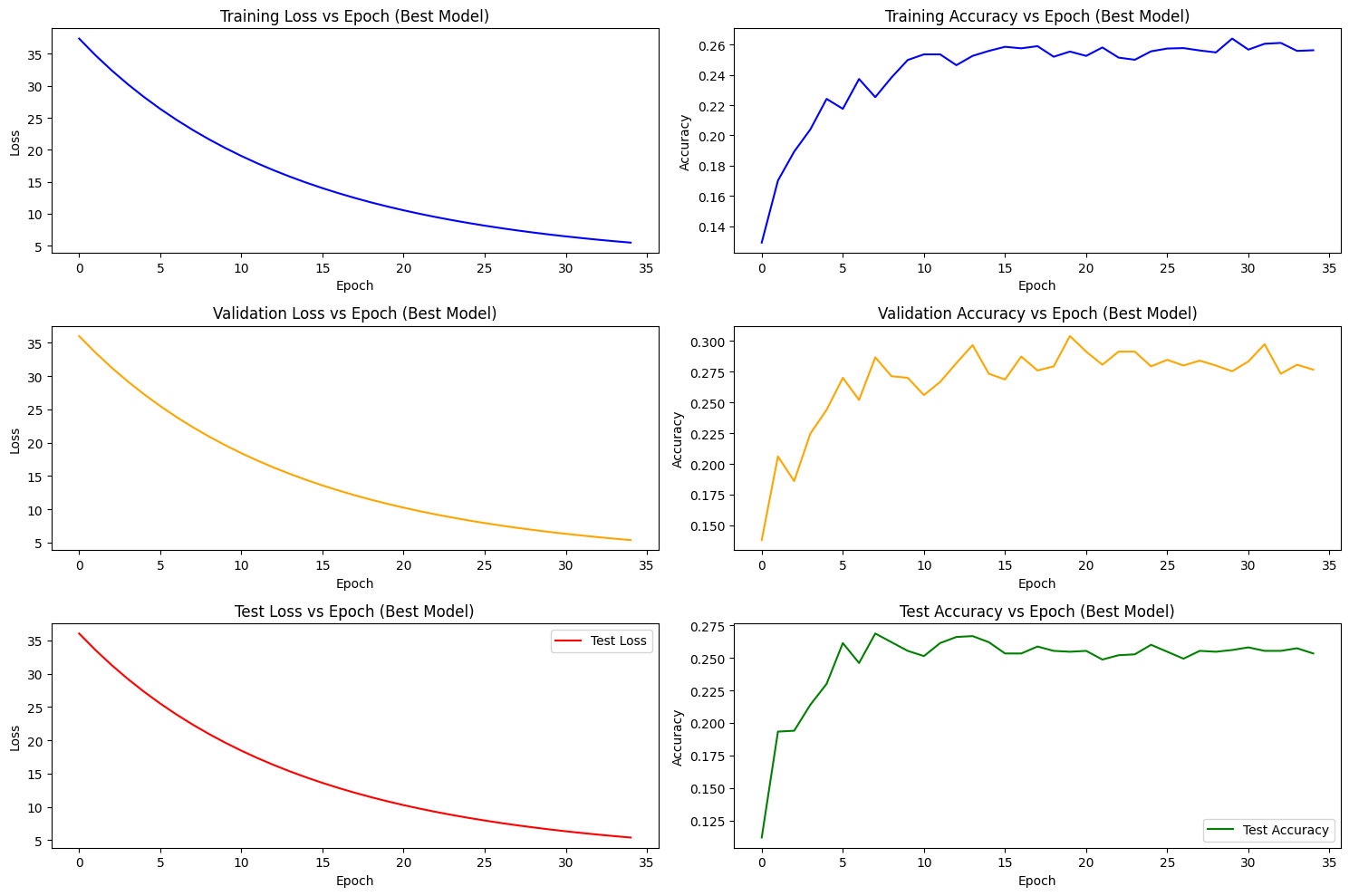
Time taken by the code to run: 1748.88 seconds

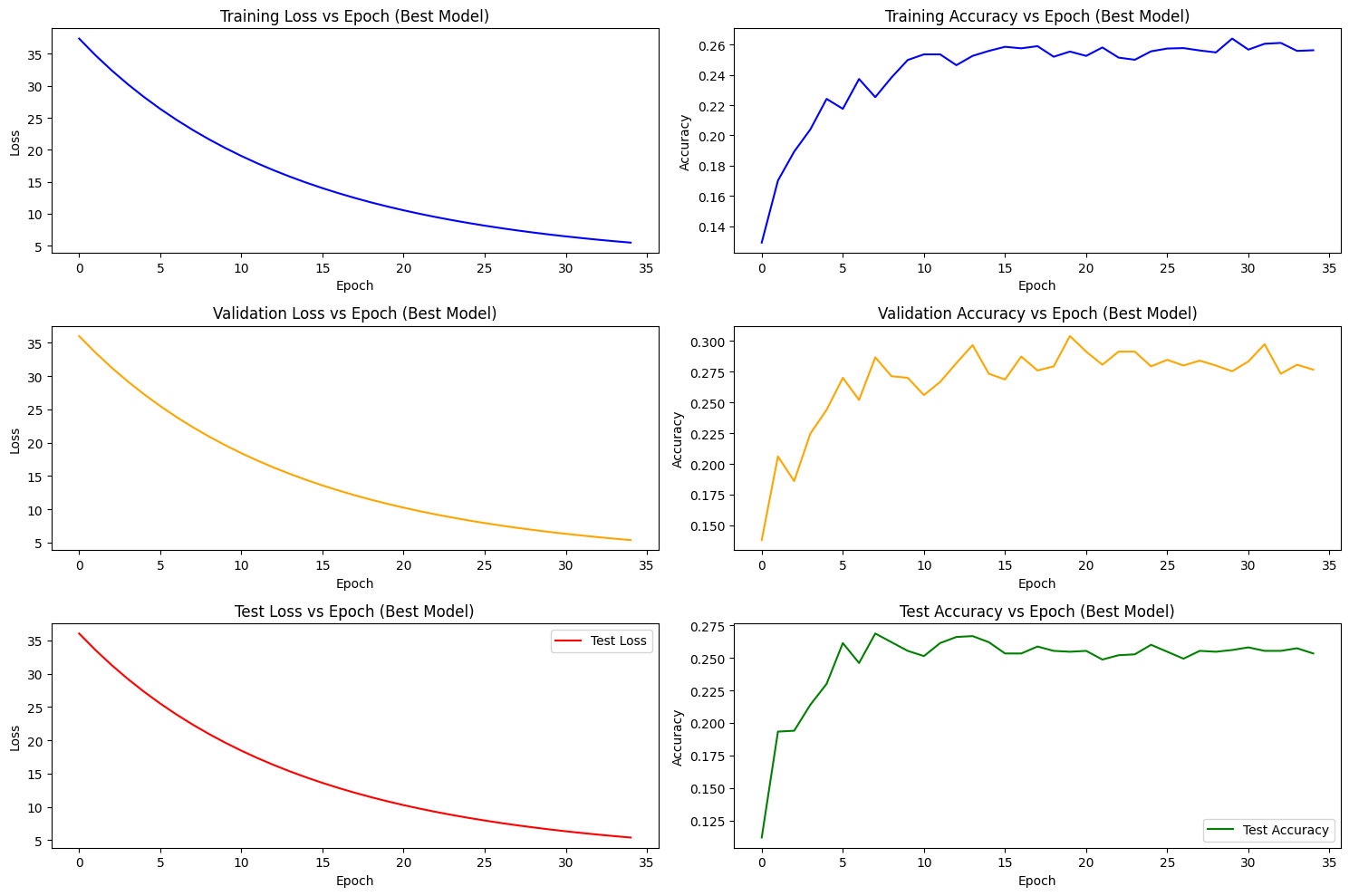
Test Loss: 4.846

Test Accuracy: 0.2682 = 26.82%

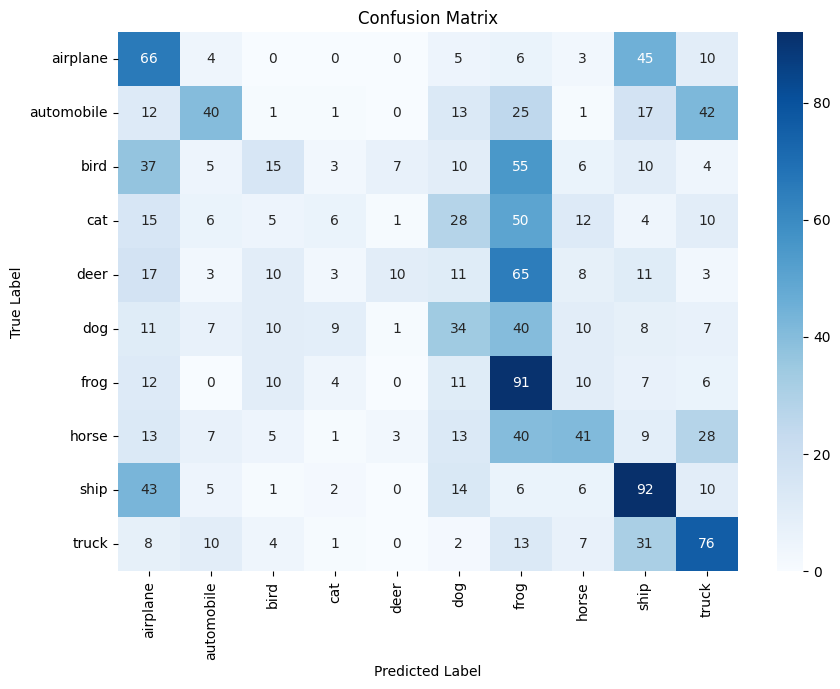
**Plots for Random Search Optimization**

****

****

****

**Results**

****

**Precision, Recall and F1-Score Table:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** |
| **airplane** | 0.28 | 0.47 | 0.35 |
| **automobile** | 0.46 | 0.26 | 0.33 |
| **bird** | 0.25 | 0.10 | 0.14 |
| **cat** | 0.20 | 0.04 | 0.07 |
| **deer** | 0.45 | 0.07 | 0.12 |
| **dog** | 0.24 | 0.25 | 0.24 |
| **frog** | 0.23 | 0.60 | 0.34 |
| **horse** | 0.39 | 0.26 | 0.31 |
| **ship** | 0.39 | 0.51 | 0.45 |
| **truck** | 0.39 | 0.50 | 0.44 |

**Discussions**

**Plots:**

When the accuracy curve plateaus after just 7-8 epochs, it signifies that the model is no longer learning significant patterns from the data using the hyperparameters chosen by the random search method. This plateau can be indicative of several issues:

1. **Suboptimal Hyperparameters**: The random search method may have settled on a set of hyperparameters that are not conducive to further improving the model's performance on this dataset.
2. **Overfitting**: The model might be too complex, fitting closely to the training data but failing to generalize well to unseen data. This is evident when training accuracy continues to rise, but validation accuracy remains stagnant or even decreases.
3. **Learning Rate Issues**: The learning rate might be too high, causing the model to miss the optimal point. Conversely, a very low learning rate can cause the model to converge too slowly, appearing as a plateau.
4. **Data Limitations**: The available data might not have enough variability or features for the model to further improve its learning after a certain point.

**Precision, Recall and F1-Score:**

1. **Performance Variation**: The random search hyperparameter optimization model displays varied results across the CIFAR-10 dataset classes.
2. **Strength in Specific Classes**: Classes such as "ship", "truck", and "horse" exhibit higher precision, recall, and F1-score values, suggesting better model accuracy and recall for these categories.
3. **Challenges in Animal Classes**: Lower performance metrics for the "bird", "cat", and "deer" classes indicate the model's difficulty in accurately distinguishing these categories due to the presence of similar features between them.
4. **High Recall for "Frog"**: Despite its average precision, the "frog" class has a high recall. This implies that the model often classifies various instances as "frog", potentially leading to misclassifications.
5. **Implication of Disparities**: The evident performance differences across classes highlight areas where the model might require further refinement.

**Bayesian Optimization (BO)**

Bayesian Optimization is a probabilistic model-based optimization technique designed for optimizing expensive-to-evaluate and noisy objective functions. It is particularly effective for high-dimensional global optimization problems.

1. **Surrogate Models**:
   * **Gaussian Process (GP)**: GP is a non-parametric model used to represent the objective function. It provides a posterior distribution over functions, which is used to model the function with uncertainty. Anisotropic kernels in GP can capture varying degrees of relevance for different input dimensions, making it robust and versatile.
   * **Random Forest (RF)**: RF is an ensemble learning method that fits multiple decision trees to the data, providing improved accuracy and control over-fitting. It is distribution-free and efficient, making it a strong alternative to GP.
2. **Acquisition Functions**:
   * **Probability of Improvement (PI)**: PI selects the next point where there is a high probability of observing improvement over the current best-known value.
   * **Expected Improvement (EI)**: EI quantifies the expected amount by which a sample might improve over the current best-known value, guiding the search to regions with higher uncertainty and potential improvement.
   * **UCB (Upper Confidence Bound)**: UCB balances exploration and exploitation by considering both the predicted value and uncertainty at each point.

**GPEI (Gaussian Process Expected Improvement)**

GPEI consists of a Gaussian Process surrogate model with the Expected Improvement as its acquisition function. It is one of the most popular choices for hyperparameter optimization due to its ability to efficiently explore the hyperparameter space while considering uncertainty in predictions. It is based on Gaussian Processes (GP), which are used as surrogate models for the objective function *f*(*x*). The algorithm aims to solve the minimization problem:

where *χ* is our search space in which we are finding the solution.

Following are the steps of GPEI:

1. **Surrogate Modelling with Gaussian Process regression (GP):** GP is used to model the objective function . GP is defined by a mean function and a covariance function . The predictive distribution of given data *D* is:

where:

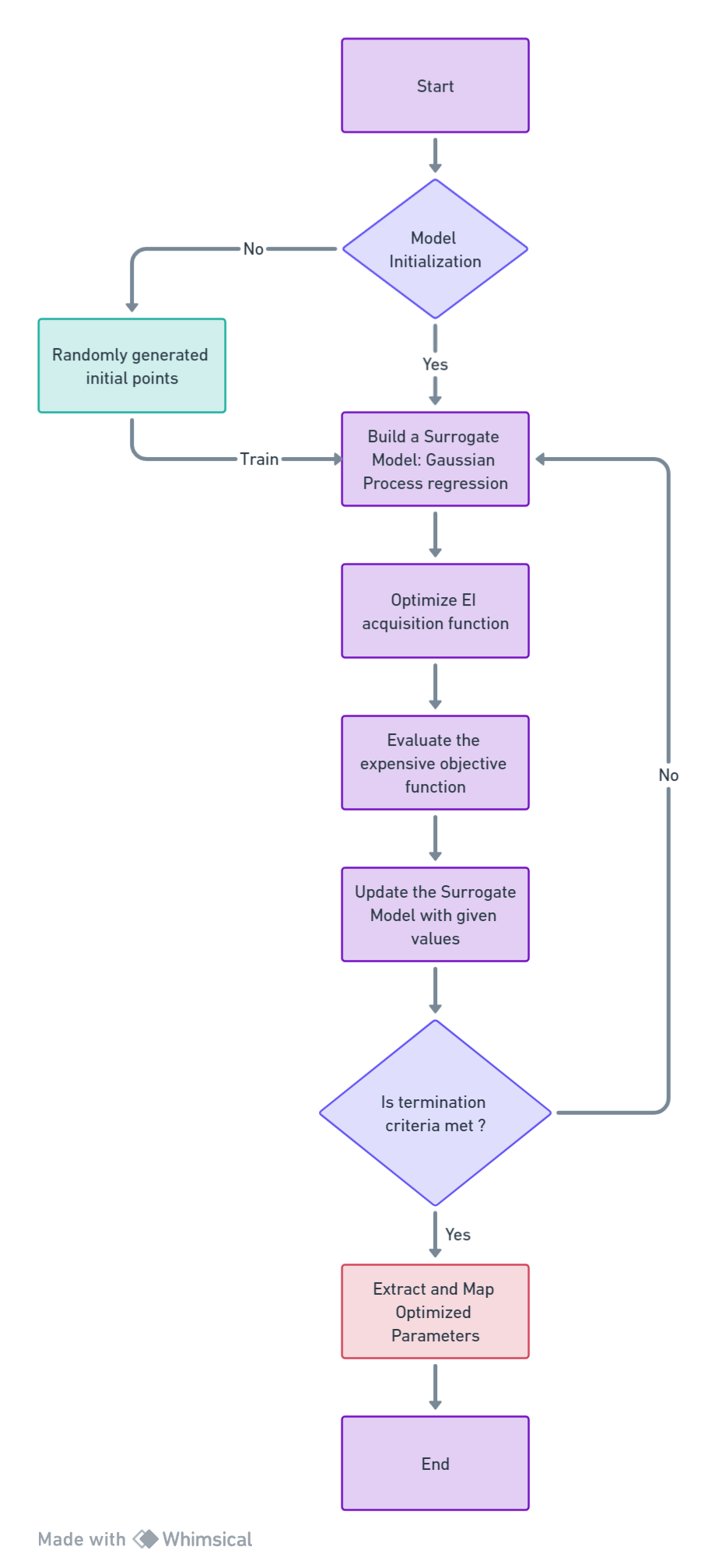
is the vector of covariances between and the training inputs, and is the matrix of covariances between the training inputs.

1. **Searching and maximizing the value acquisition Function:** The Expected Improvement acquisition function is used to decide where to sample next. Expected Improvement (EI) at a point is given by:

Where:

* is the predicted mean at .
* is the best observed value.
* is the predicted standard deviation at .
* is a small positive value (exploration-exploitation trade-off).
* and are the CDF and PDF of the standard normal distribution, respectively.

**GPEI Bayesian Optimization Flowchart**



**Results of Hyperparameter optimization using GPEI Algorithm**

|  |  |
| --- | --- |
| Hyperparameters | Optimized Value |
| Number of convolutional layers | 4 |
| Number of Filters | 256 |
| Dropout Rate | 0.7 |
| Optimizer | sgd |
| Learning rate | 0.00073 |
| Activation function | elu |
| Epochs | 42 |
| Batch size | 32 |
| Kernel size | 3 x 3 |
| Regularization Lambda | 0.00047 |

Maximum Training accuracy: 0.745 = 74.5%

Time taken by the code to run: 4904.44 seconds

Test Loss: 1.1767

Test Accuracy: 0.5884 = 58.84%

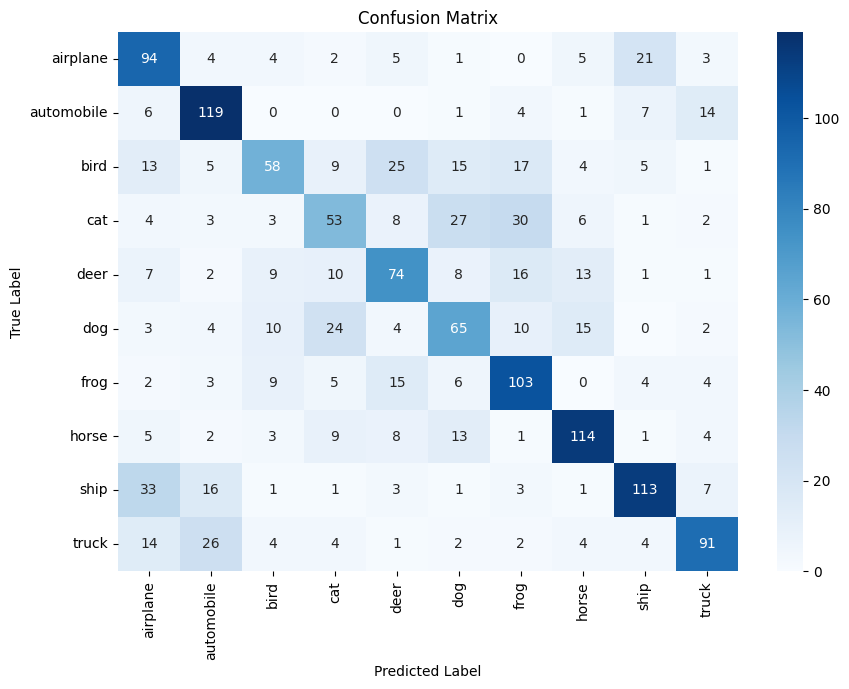
**Plots for GPEI Optimization**

****

****



**Results**



**Precision, Recall and F1-Score Table:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** |
| **airplane** | 0.52 | 0.68 | 0.59 |
| **automobile** | 0.65 | 0.78 | 0.71 |
| **bird** | 0.57 | 0.38 | 0.46 |
| **cat** | 0.45 | 0.39 | 0.42 |
| **deer** | 0.52 | 0.52 | 0.52 |
| **dog** | 0.47 | 0.47 | 0.47 |
| **frog** | 0.55 | 0.68 | 0.61 |
| **horse** | 0.70 | 0.71 | 0.71 |
| **ship** | 0.72 | 0.63 | 0.67 |
| **truck** | 0.71 | 0.60 | 0.65 |

**Discussions**

Plots:

The testing accuracy shows a general increasing trend with only a few minor dips and is accompanied by a low loss, it provides several insights into the model's performance and characteristics:

1. **Generalization**: The upward trend in testing accuracy suggests that the model is recognizing patterns beyond the training data and is capable of making predictions on new, unseen data.
2. **Learning Dynamics**: Occasional decreases in accuracy can signal the model's adjustments to certain data patterns. The following upward trend indicates successful learning and adjustment.
3. **Overfitting Assessment**: A low loss on the test set indicates alignment between the model's predictions and actual outcomes. Together with the upward accuracy trend, this suggests a balance in the model's fit to the data.
4. **Convergence Behaviour**: The trend of increasing accuracy and low loss points to a stable model training process, implying that hyperparameters such as the learning rate are facilitating convergence.
5. **Optimization Potential**: If the accuracy trend is still upward in the later epochs, there might be room for further model enhancement, be it through extended training, architecture modifications, or hyperparameter adjustments.

Precision, Recall and F1-Score:

1. **Model Strength on Specific Classes**: The Bayesian optimization model shows pronounced performance on classes like "automobile", "horse", "ship", and "truck", as evidenced by their higher precision, recall, and F1-scores. This suggests that the model can consistently predict and recall these categories.
2. **Challenges with Animal Classes**: Lower metrics for "bird", "cat", and "dog" indicate potential challenges in the model's prediction and recall capabilities for these classes. This might result from similar features between these classes.
3. **Possible Feature Overlap**: The overlap in features between classes like "cat" and "dog" could be contributing to model confusion, leading to misclassification.
4. **Dataset Representation**: The representation of classes in the dataset, like "bird", might affect their classification performance. If certain features or variations are underrepresented, it can influence the model's ability to generalize for that class.
5. **Insights into Multiclass Classification**: The varied performance across classes provides insight into the model's strengths and weaknesses when handling multiclass data. Clear and distinct features associated with each class, along with their representation in the dataset, play a pivotal role in model learning.

**Multilevel Evolution Optimization (MLEO)**

Multilevel Evolutionary Optimization Algorithm (MLEO) is a genetic algorithm that is global search-based, drawing inspiration from natural phenomena. It is recognized for its robustness and independence from specific problems, making it a versatile tool for parameter optimization. MLEO here is a multilevel genetic algorithm because it incorporates both within-group and between-group dynamics, making it an exhaustive optimization algorithm. Through applications of various processes like selection, crossover, mutation, migration, and regrouping processes, it searches through the search space efficiently, fostering cooperation and competition among individuals and groups to find optimal solutions to complex optimization problems.

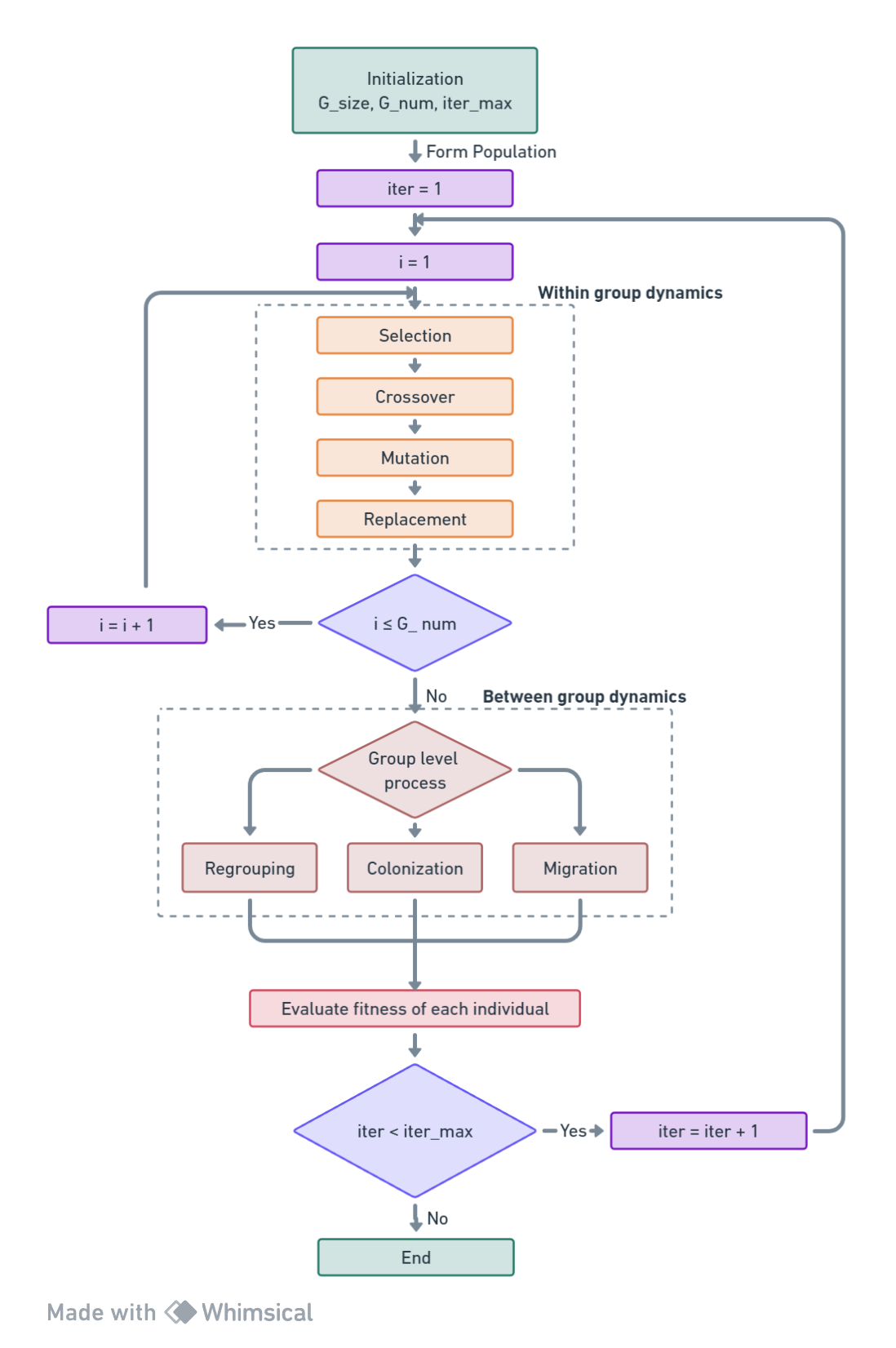
**Within-Group Dynamics:** Within-group dynamics are the fast evolutionary processes occurring within each group during each cycle of the MLEO. The dynamics are:

* **Individual Selection:** A process where two individual solutions are chosen as parents from the same group, based on their fitness values, to reproduce the next generation of solutions. The individual having a fitness of has a chance of being selected as a parent is given as:

* **Crossover:** A genetic operator used to combine the genetic information of two parents to reproduce new offspring. This process helps in coming up with potentially better solutions, known as offspring.
* **Mutation:** It is a genetic operator which introduces small changes in the candidates' features, promoting diversity and preventing stagnation in solution within the group and helping to escape being stuck in local optima.
* **Fitness Function:** The fitness function evaluates the performance or suitability of individuals within the population. It evaluates how close a solution is to the optimum and quantifies the solution’s goodness or fitness.

**Between-Group Dynamics:** Between-group dynamics are much slower evolutionary processes that occurs between the groups. The dynamics are:

* **Extinction-Colonization Process:** It is a process by which solutions can migrate between groups, diversifying and helping in global optimization. One group is chosen as a colonist, and the another is marked for extinction. New offspring is generated through crossover and added to the colonist group. The colonist group is randomly split into two, with one replacing the original colonist and the other replacing part of the extinct group based on the fitness of individuals.
* **Migration Process:** This process focuses on cooperation among groups through bidirectional migration channels. Individuals are selected for migration based on a roulette wheel method that evaluates their relative fitness. Those with lower fitness have a higher chance of migrating to other groups.
* **Regrouping Process:** Group boundaries are dissolved, and individuals are mixed or regrouped within the population to make sure that the population is diverse and to prevent premature convergence.

**Flowchart for Multilevel Evolution Optimization**

**Results of Hyperparameter optimization using MLEO Algorithm**

|  |  |
| --- | --- |
| Hyperparameters | Optimized Value |
| Number of convolutional layers | 3 |
| Number of Filters | 76 |
| Dropout Rate | 0.375 |
| Optimizer | adagrad |
| Learning rate | 0.0031 |
| Activation function | tanh |
| Epochs | 50 |
| Batch size | 64 |
| Kernel size | 3 x 3 |
| Regularization Lambda | 0.0166 |

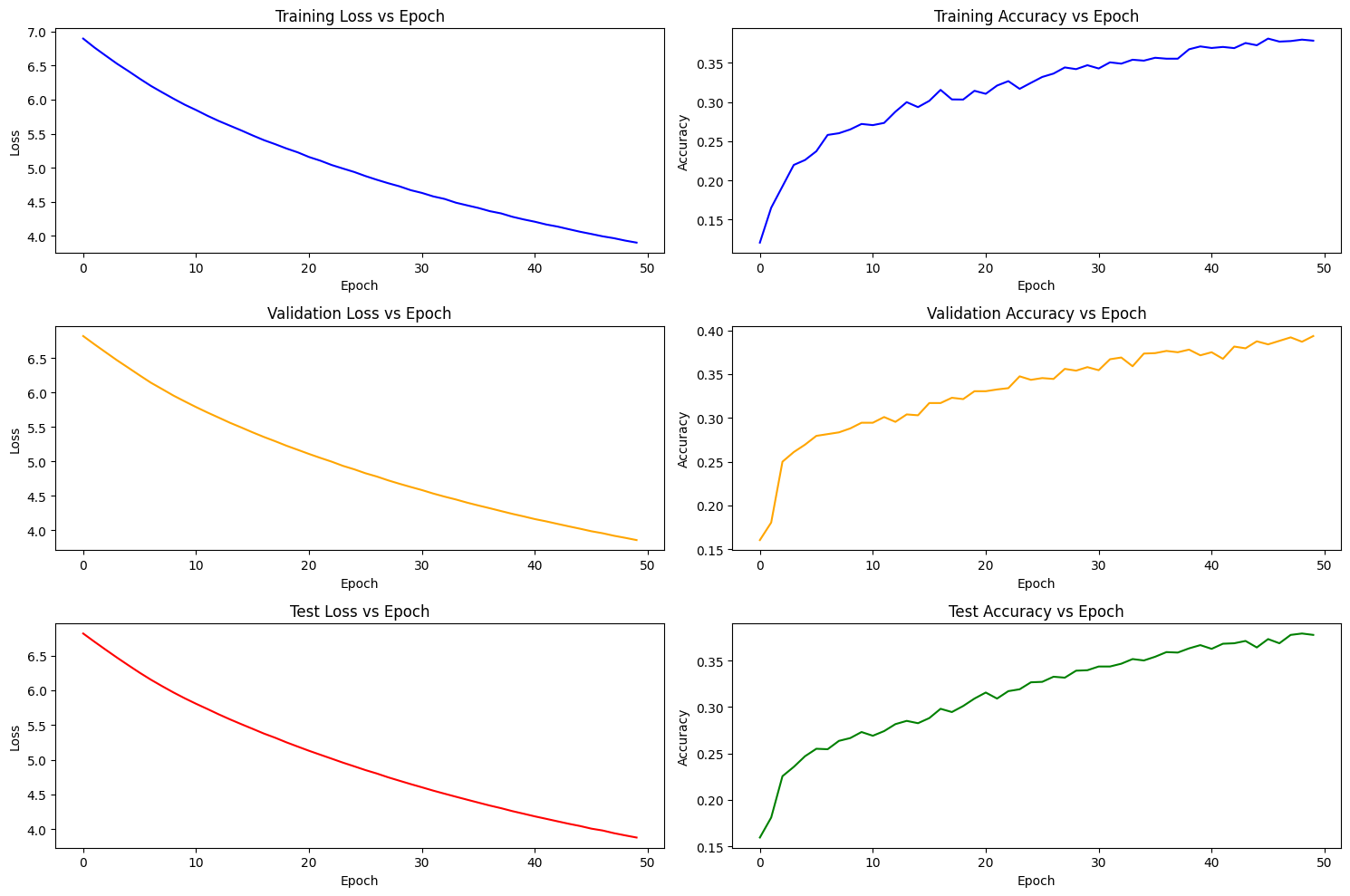
Maximum Training accuracy: 0.3815 = 38.15%

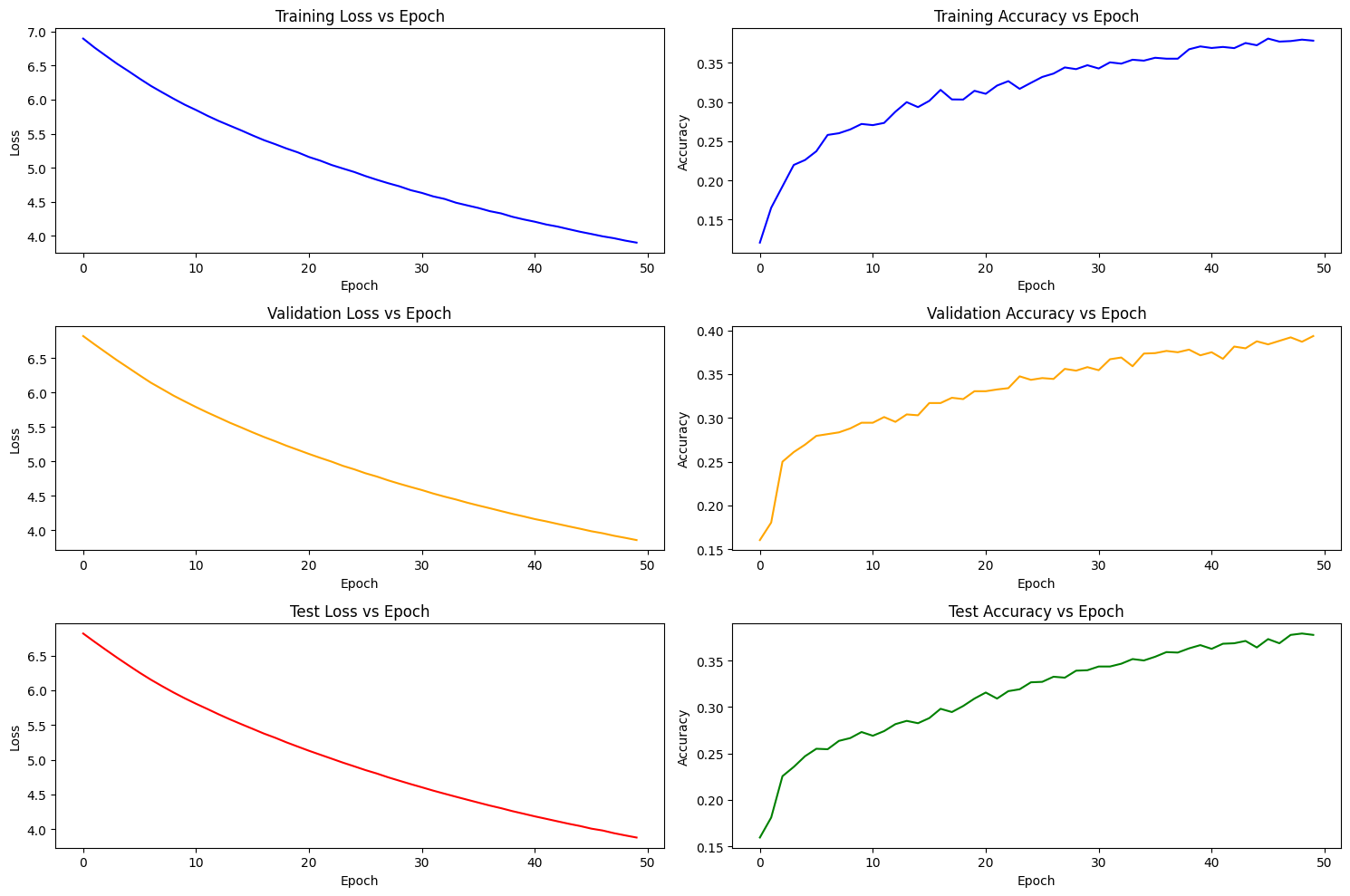
Time taken by the code to run: 7015.35 seconds

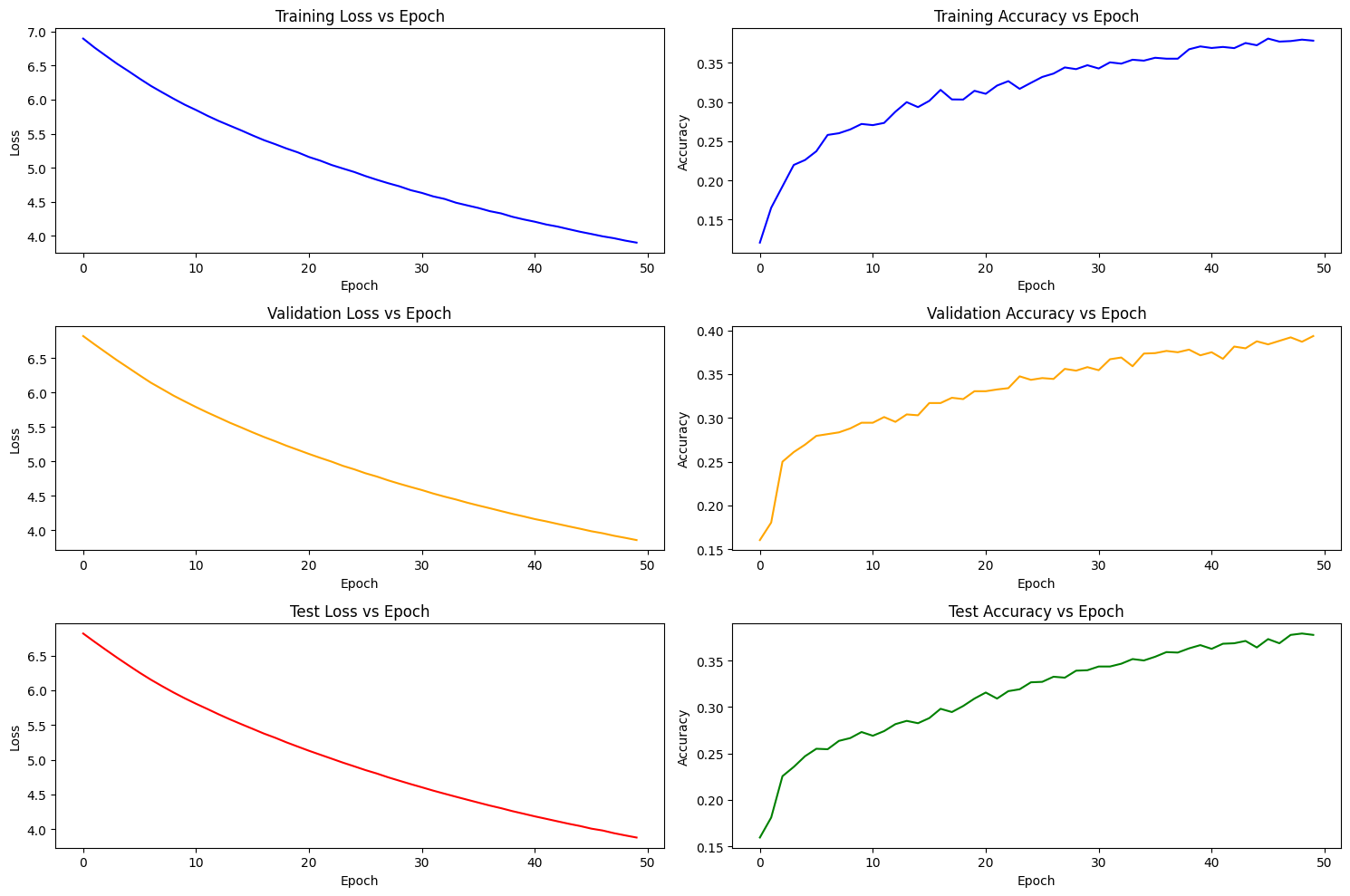
Test Loss: 3.84

Test Accuracy: 0.386 = 38.6%

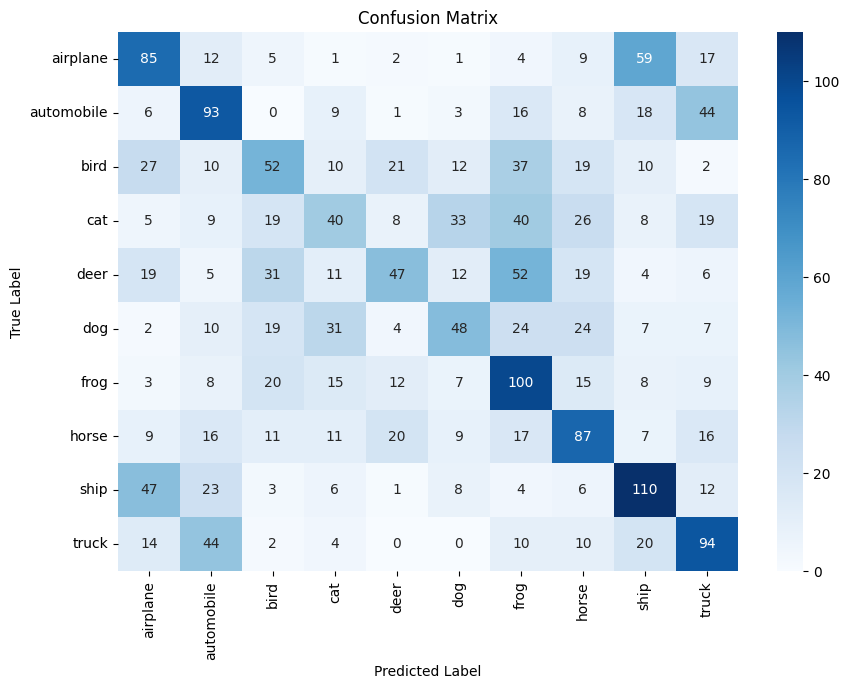
**Plots for MLEO algorithm**

****

****

****

**Results**

****

**Precision, Recall and F1-Score Table:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** |
| **airplane** | 0.39 | 0.44 | 0.41 |
| **automobile** | 0.40 | 0.47 | 0.43 |
| **bird** | 0.32 | 0.26 | 0.29 |
| **cat** | 0.29 | 0.19 | 0.23 |
| **deer** | 0.41 | 0.23 | 0.29 |
| **dog** | 0.36 | 0.27 | 0.31 |
| **frog** | 0.33 | 0.51 | 0.40 |
| **horse** | 0.39 | 0.43 | 0.41 |
| **ship** | 0.44 | 0.50 | 0.47 |
| **truck** | 0.42 | 0.47 | 0.44 |

**Discussions**

**Plots:**

The testing accuracy shows a general increasing trend with only a few minor dips and is accompanied by a low loss, it provides several insights into the model's performance and characteristics:

1. **Generalization**: The model quickly captures the fundamental patterns in the data within the initial 4-5 epochs. The subsequent gradual increase in accuracy over the next 45 epochs implies that the model further refines its understanding and captures more intricate patterns and nuances.
2. **Learning Dynamics**: The sharp increase in testing accuracy during the initial epochs indicates that the model effectively overcomes its random initialization, adjusting its weights to better represent the data.
3. **Overfitting Assessment**: The stable testing loss, coupled with an increasing testing accuracy, can be indicative of the model's balance in fitting the data. While it is learning and improving its predictions, the stable loss suggests it is not becoming overly confident, mitigating the risk of overfitting
4. **Convergence Behaviour**: The stability in the testing loss around 3.8, even with increasing accuracy, indicates a stable convergence behaviour during training, suggesting that the learning rate and other training-related hyperparameters are facilitating smooth convergence without excessive oscillations.
5. **Optimization Potential**: The model's testing accuracy, still rising towards the latter epochs, hints at potential for further improvements. This could be achieved by more extended training, refining the architecture, or hyperparameter tuning.

**Precision, Recall and F1-Score:**

1. **Model Strength on Vehicles**: The model demonstrates relatively stronger performance on vehicle classes, namely "ship", "truck", "airplane", and "automobile", with higher precision, recall, and F1-scores.
2. **Challenges with Animal Classes**: There is a noticeable decline in metrics for classes like "cat", "bird", and "deer", indicating the model's struggles in distinguishing between certain animal categories.
3. **Feature Distinction**: The more effective capture of features from vehicle classes suggests that these objects might have distinct features that the model can easily identify, while animal classes might possess overlapping or similar features that make differentiation more challenging.
4. **Inherent Class Similarities**: The difficulty in distinguishing between animal classes might be due to inherent similarities in features between them, which are harder for the model to differentiate.

**Combination of MSMLEO\_GPEI**

The term "MSMLEO" is used to highlight the incorporation of the multiscale approach into the original MLEO algorithm when combined with GPEI. The "MS" in MSMLEO stands for "Multiscale," emphasizing the multiscale nature of the optimization process in the combined algorithm. GPEI is a Bayesian optimization method that is effective in optimizing continuous hyperparameters. In the hybrid approach, MSMLEO focuses on optimizing discrete hyperparameters, while GPEI optimizes continuous ones.

This multiscale strategy introduces grids representing the hyperparameter space at various scales or levels of granularity, facilitating a sequential and hierarchical optimization process. Starting at the coarsest scale, the algorithm identifies broad regions of interest in the hyperparameter space and progressively refines the search at finer scales for precise optimization. This approach not only provides a global perspective through coarse-scale optimization but also allows for detailed, localized search within promising regions through fine-scale optimization. While MSMLEO primarily focuses on optimizing discrete hyperparameters like the number of layers or filter sizes in neural networks, GPEI excels at tuning continuous hyperparameters, such as learning rates. The integration of the two algorithms enables coordinated optimization of both discrete and continuous hyperparameters, with MSMLEO’s multiscale multilevel approach working seamlessly with GPEI’s Bayesian optimization.

Advantages of this hybridized MSMLEO\_GPEI approach over MLEO and GPEI:

* The combination of MLEO and GPEI into MSMLEO is crucial for efficient hyperparameter tuning in machine learning models, addressing the need for a robust method that can optimize both discrete and continuous hyperparameters.
* The multiscale approach facilitates hierarchical optimization. The coarse-scale optimization provides a global perspective, guiding the search towards promising regions, while the fine-scale optimization conducts a more detailed, localized search within those regions.

**Conclusions**

Hyperparameter optimization is a crucial step in the model training process, significantly impacting the performance and efficiency of learning algorithms. This report has explored the importance of selecting ideal hyperparameters, which are external configurations not derived from the data but instrumental in controlling the learning process.

Various techniques for hyperparameter optimization have been discussed, each with its unique approach and application. Grid Search offers a straightforward method, meticulously exploring every possible combination of predefined hyperparameter values. While exhaustive, this technique is computationally intensive and time-consuming, often impractical for high-dimensional spaces.

Random Search introduces an element of randomness, selecting hyperparameter combinations at random. This approach is more computationally efficient than Grid Search, providing a practical alternative for exploring the hyperparameter space swiftly and effectively. Although a popular approach, it seemed to plateau early during training, suggesting potential suboptimal hyperparameter selections or overfitting tendencies. Its class-wise analysis also pointed out challenges in differentiating between certain animal classes, possibly due to feature overlaps or data representation issues.

Furthermore, Bayesian Optimization was highlighted as a sophisticated technique, constructing a probabilistic model of the objective function to intelligently select promising hyperparameter combinations. This method is particularly beneficial for optimizing expensive-to-evaluate functions, balancing exploration, and exploitation to navigate towards optimal solutions efficiently. It showcased an upward trend in testing accuracy and a stable loss, indicative of its robust generalization and effective learning dynamics. Despite this, it still faced challenges in distinguishing between certain animal classes, much like the Random Search method.

Lastly, Multilevel evolution optimization is shown as a robust and exhaustive optimization method which includes both in-group and between-group dynamics which helps in finding the global optimum values and avoids premature convergence, ensuring a comprehensive search of the solution space. It exhibited similar positive trends in testing accuracy but a higher loss showing it was mitigating the overfitting risk. It effectively recognized vehicle classes, but, like the other two methods, had difficulties with certain animal classes.

In conclusion, the meticulous selection and optimization of hyperparameters are paramount for developing robust and efficient learning algorithms. Researchers and practitioners should consider the unique characteristics and requirements of their problems when selecting an optimization technique, as the right method can significantly enhance model performance and training efficiency. Future work in this domain may explore the integration and comparison of different optimization techniques, as well as the development of novel methods to address the ever-evolving challenges of hyperparameter optimization in machine learning.

# References

[1] Cui, H., & Bai, J. (2019). A new hyperparameters optimization method for convolutional neural networks. *Pattern Recognition Letters*, *125*, 828-834.

[2] Yang, L., & Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, *415*, 295-316.

[3] Kim, J., & Choi, S. (2021). On local optimizers of acquisition functions in bayesian optimization. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part II* (pp. 675-690). Springer International Publishing.

[4] Lei, B., Kirk, T. Q., Bhattacharya, A., Pati, D., Qian, X., Arroyave, R., & Mallick, B. K. (2021). Bayesian optimization with adaptive surrogate models for automated experimental design. *Npj Computational Materials*, *7*(1), 194.

[5] Akbari, R., & Ziarati, K. (2011). A multilevel evolutionary algorithm for optimizing numerical functions. *International Journal of Industrial Engineering Computations*, *2*(2), 419-430.