Optimizing Flower Classification Models: A Comparative Analysis of Bayesian Optimization and Random Search for Hyper parameter Tuning

Naresh Dembla[1], Ravindra Yadav[2*] ,Urvashi sharma[3] , Devendra Singh[4]

[1]Assistatnt Professor, IIPS, DAVV, Indore (M.P), INDIA
[2] Assistant Professor IET, DAVV , Indore(M.P). , INDIA
[3,4] Assistant Professor ,AITR , Indore(MP) , INDIA,
nareshdembla@gmail.com
*ryadav@ietdavv.edu.in
urvashisharmazero9@gmail.com
devendrasingh@acropolise.in

Abstract. Understanding plant varieties, ecosystem health, and agricultural methods are made easier with the help of flower categorisation, which is essential to botanical study, environmental monitoring, and agriculture. The effectiveness of automating flower classification has been greatly attributed to recent developments in machine learning, particularly in deep neural networks. But in order to maximise model performance, hyper parameter optimisation is crucial. By utilising state-of-the-art hyper parameter optimisation approaches, this research seeks to improve the precision and efficacy of floral categorisation systems. The work addresses issues such dataset anomalies, such as photos of flowers in unusual locations, by utilising a Kaggle dataset with 104 flower species. In order to maximise the macro F1 score—a crucial metric for multiclass classification—the study contrasts random search versus Bayesian optimisation for hyper parameter tweaking. The study obtained an F1 score of 0.966 with random search using the Swin Transformer model, as opposed to 0.9285 with Bayesian optimisation. With batch sizes optimised for TPU utilisation, the dataset was divided into 77.46% for training and 22.54% for validation.

Keywords: Flower Classification, Swim Transformer, Random Search, Bayesian Search, Hyper Parameter.

1 Introduction

Flower classification plays a crucial role in various domain, ranging from botanical research to environmental monitoring and agriculture .The ability to accurately

S. Bhalerao et al. (eds.), *Proceedings of the International Conference on Recent Advancement and Modernization in Sustainable Intelligent Technologies & Applications (RAMSITA-2025),* Advances in Intelligent Systems Research 192,
https://doi.org/10.2991/978-94-6463-716-8_16

identify and classify flower species is fundamental for understanding plant diversity, ecosystem health, and agricultural practices .In recent years, machine learning techniques, coupled with deep neural network ,have demonstrated remarkable success in automating the flower classification process. However, achieving optimal performance in machine learning models require careful tuning of hyper parameters, which critical configuration is setting influencing the model's behavior and performance.Propsed work focus on leveraging advanced hyper parameter optimization technique to enhance the accuracy and efficiency of flower classification models.

2 Related Work

The authors have examined and illustrated the most current advancements in deep learning techniques, including CNN and CNN's transfer learning. A suggested prototype CNN model architecture and a transfer learning strategy are also evaluated for flower classification on publically accessible flower datasets using the VGG16, MobileNet2, and Resnet50 architectures [1]. The Swin (Shifted Window) Transformer model is an examination of a variant of the vision transformer. This is a representation-computed hierarchical transformer model with shifted windows. The performance of this transformer for three distinct remote sensing datasets—EuroSat, NWPU-RESISC45, and AID—was thoroughly examined in the results. Results show that for accurate remote sensing picture classification, the Swin architecture performs better than the state-of-the-art methods currently in use [2]. In the context of machine learning, transfer learning refers to using the output from many DNN applications. This paper examines, using four publicly available datasets, the effects of four distinct transfer learning models for deep neural network-based plant categorization. Our experimental investigation shows that transfer learning may enhance low-performing plant classification models and offer significant advantages for automated plant identification [3].The first insect dataset utilized was the National Bureau of Agricultural Insect Resources (NBAIR) dataset, which has 40 classes of field crop insect photos. The second and third datasets (Xie1, Xie2), on the other hand, had 24 and 40 classes of insects, respectively. In order to classify insects, the suggested model was assessed and contrasted with pre-trained deep learning architectures including AlexNet, ResNet, Google Net, and VGGNet. The pre-trained models were refined through the use of transfer learning [4]. This paper proposes an optimization strategy for Hyperparameter optimization of deep CNN for plant species identification, which is based on the artificial bee colony (ABC) algorithm and is named the optimum deep CNN (ODC) classifier. It is used on a pre-made leaf dataset called Folio, which has #637 photos representing 32 distinct plant species. To increase the ODC classifier's effectiveness, the pictures undergo a variety of image preparation techniques, including segmentation, augmentation, and scaling. Consequently, the dataset is expanded from #637 to #15,288 leaf pictures, of which #12,103 are set aside for the training phase and the remaining number for ODC testing. Furthermore, in both the optimization and classification stages, a validation procedure is carried out on 20 percent of the training dataset in addition to the training phase[5]. This work made use of a diverse picture collection of plant leaves with 12 different crops in 22

different categories. Training a deep learning model becomes increasingly difficult and demanding due to the intra- and inter-class variances in the training dataset. Numerous deep neural networks have been thoroughly examined using various combinations of optimizers and learning rates. Furthermore, a thorough examination of the trained model in various statistical parameters has been conducted by testing on distinct test pictures and five-fold cross-validation. With unseen pictures with intra- and inter-class changes, the suggested method improved the results to an average cross-validation accuracy of 98.68% and an average test accuracy of 97.69% [6]. Deep learning (DL) techniques, particularly those based on convolution neural networks (CNN), are currently being used extensively in the categorization of plant diseases. They embody cutting-edge technology in this domain and have resolved, at least in part, the issues with conventional categorization techniques. We examined the most recent CNN networks relevant to the categorization of plant leaf diseases in this work. We outlined the key DL concepts for classifying plant diseases. Furthermore, we outlined the primary issues with CNN's plant disease categorization system and their associated fixes. We also spoke about the route that plant disease categorization would take in the future[7] While accuracy, precision, recall, and F1-score are the most often used assessment measures, using them alone has several drawbacks[8].

3 Hyperparameter optimization

Table 1 describes the strength and limitation of both optimization techniques, adopted in the proposed research work in terms of efficiency, applicability, and ease of use .The analysis of Random search and Bayesian hyper parameter optimization technique has been done for the classification of flowers based on the different factors.

Table 1: A comparative study of applied hyper parameter tuning methods [8] [9] [10]

Aspect	Random Search	Bayesian Optimization
Search Space	Discrete values for hyper parameters	Continuous search space with adaptive exploration
Hyper parameters _estimators	[50, 100, 200, 300]	Not applicable for Bayesian Optimization (continuous space)
max_depth	[None, 10, 20, 30]	Not applicable for Bayesian Optimization (continuous space)
min_samples_split	[2, 5, 10]	Not applicable for Bayesian Optimization (continuous space)
min_samples_leaf	[1, 2, 4]	Not applicable for Bayesian

bootstrap	[True, False]	Optimization (continuous space) Not applicable for Bayesian Optimization (continuous space)
Exploration Strategy	Random sampling	Adaptive sampling based on previous evaluations
Computational Efficiency	May require more iterations	Efficient in terms of the number of trials
Handling Continuous Hyper parameters	Requires discretization or sampling	Direct support for continuous hyper parameters
Parallelization	Straightforward parallelization	Efficient parallelization through surrogate modeling
Adaptability to Problem Domain	Versatile and suitable for various problems	Effective for problems with smooth and continuous search spaces
Implementation Complexity	Simple to implement	Requires integration with a specialized library (e.g., scikit-optimize)
Best Hyperparameter Retrieval	Directly obtained from `best_params_` attribute	Directly obtained from `best_params_` attribute

4 Methodology

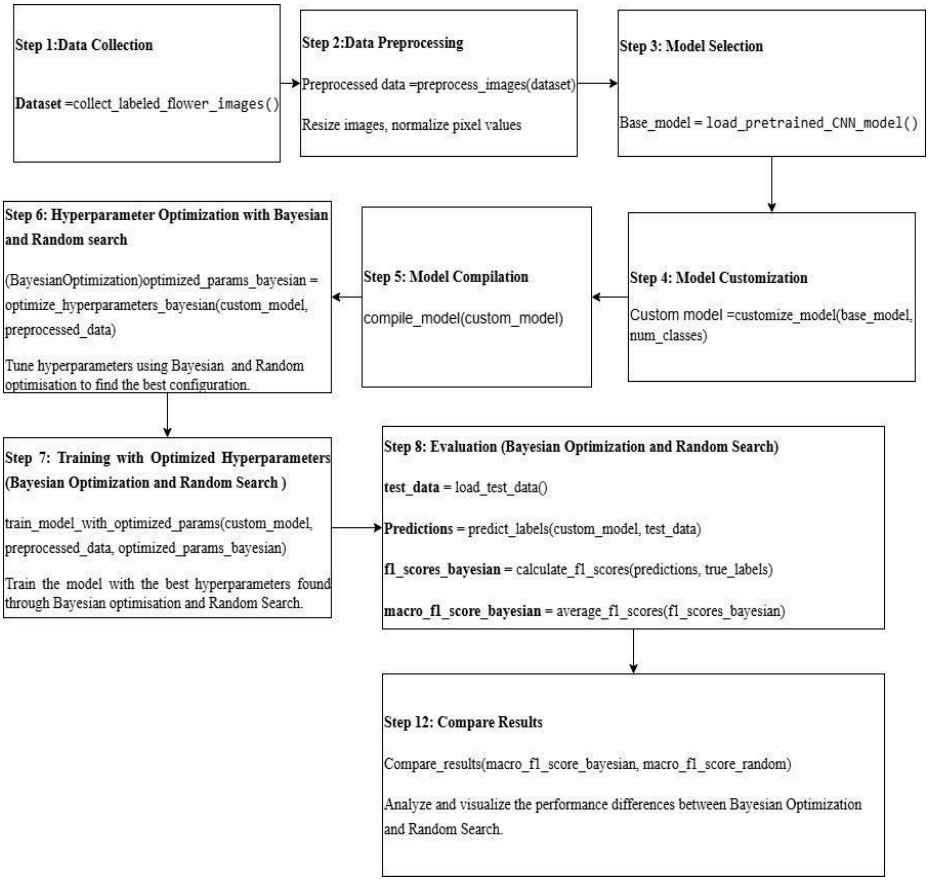

Fig. 1. Flow chart of the proposed work.

Leveraging advanced hyper parameter optimization technique to enhance the accuracy and efficiency of flower classification models.

Figure 1 describes the flow of proposed work ,which focus on flower classification using Swin transformer model .It involves collecting and preprocessing labeled flower images ,customizing the pertained model ,and optimizing hyper parameter using Bayesian optimization and Random search .The model is trained and evaluated using F1-score and the results from both the optimization techniques are compared to determine the best approach for the classification .The prosed work intended to enhance accuracy and performance by leveraging transformer based deep learning and advanced hyper parameter tuning .

5 Experimental Setup

The experimental setup shows pipeline for machine learning model intended for the classification of flowers. It manages dataset storage and retrieval using TFRecord files in Google Cloud Storage, analyses photos with dimensions [224,224] and [512, 512], and makes use of a TPU for hardware acceleration [11]. Tensor Flow 2.0 is intended to run in cloud-based environments Google Colab. Tensor Flow's AUTOTUNE is used for optimizing memory usage during data loading.

5.1 Learning rate schedule

Input:
- LR_START: Initial learning rate
- LR_MAX: Maximum learning rate after warm-up
- LR_MIN: Minimum learning rate after decay
- LR_RAMPUP_EPOCHS: Number of epochs for the learning rate warm-up
- LR_SUSTAIN_EPOCHS: Number of epochs to sustain the maximum learning rate
- LR_EXP_DECAY: Exponential decay factor for the learning rate

Output:
- Learning Rate Schedule

5.2 Algorithm (Three –Phase LR schedule)

```
Function lrfn (epoch):
If epoch < LR_RAMPUP_EPOCHS:
Lr = (LR_MAX - LR_START) / LR_RAMPUP_EPOCHS * epoch + LR_START
Elif epoch < LR_RAMPUP_EPOCHS + LR_SUSTAIN_EPOCHS:
Lr = LR_MAX
Else:
Lr= (LR_MAX - LR_MIN) * LR_EXP_DECAY^ (epoch - LR_RAMPUP_EPOCHS
-  LR_SUSTAIN_EPOCHS) + LR_MIN
Return lr
Function get_lr_callback (plot_schedule=False):
If plot_schedule:
Rng = [i for i in range (25 if EPOCHS < 25 else EPOCHS)]
y = [lrfn(x) for x in rng]
plot_learning_rate_schedule (rng, y)
Return LearningRateScheduler (lrfn, verbose=0)
```

The experimental setup section 5.1 communicates the approach used in the proposed work t train the neural networks,highlighting the importance of dynamic learning rate adjustment in potentially enhancing model trining outcomes.

The Three phase LR schedule algorithm in section 5.2 perform the dynamic adoptation ,balance exploratin(high LR) and exploitatiom(low LR) during training .Ramp up duration ,sustain period and decay rate can be tune for specific task thus giving the flexibility. Figure 2 describes the different stages of the learning rate schedule, from small starting value which ensure stable learning of the model,

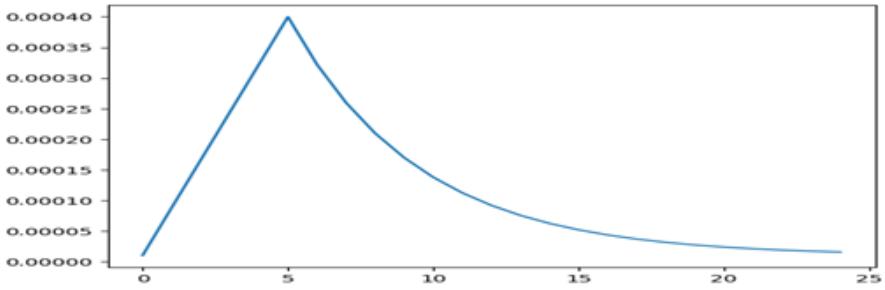

Fig.2. Learning rate schedule: 1e-05 to 0.0004 to 1.56e-05

a significant jump of 0.0004 for rapid learning in the early stages, and a fine tuning phase show the reduction of step size to stabilize training and present overshooting.

6 Dataset

12753 training images, 3712 validation images, 7382 unlabeled test images [11].Split Ratio= Number of Validation Images Total Number of Training and Validation Images Split Ratio=3,712/(12,753+3,712)Split Ratio≈22.54% for validation and 77.46% for training The batch size is defined based on whether a TPU (Tensor Processing Unit) is being used or not. If TPU is enabled (strategy.num_replicas_in_sync is greater than 1), the batch size is set to 16 times the number of replicas. If TPU is not used, the batch size is set to 16.Printed Training Data Shapes: Each batch in the training dataset contains 128 images. Each image has a shape of (224, 224, 3) and (512, 512, 3), indicating a 224x224, 512x512 RGB image. Labels are one-dimensional arrays with a shape of (128,).Printed Training Data Label Examples: Examples of label arrays for the last batch in the loop were printed. Labels are integers representing the classes/categories of the corresponding images. Figure 3 describes the random sample of the input images taken for the current work.

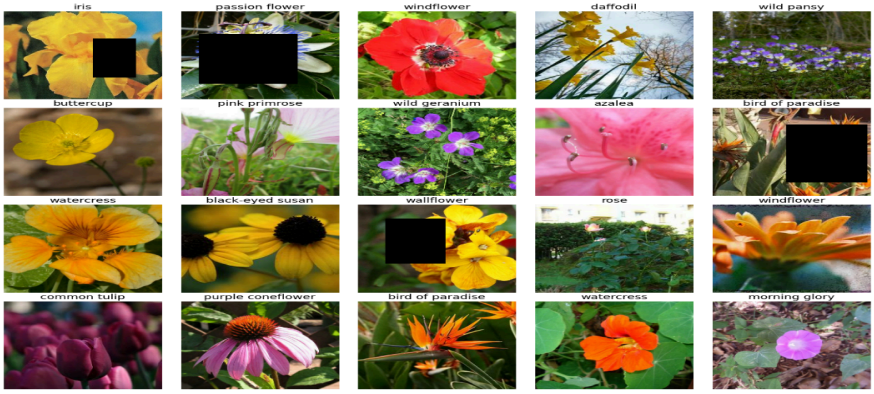

Fig.3. Random Samples of the input images.

7 Hyper Parameter Configuration for swin Transformer Model.

Table 2 represents the two different variation setup for the swin transformer model training with different input size of the flower images and the hyper parameters .The variations used to experiment model performance on different image size to optimize for the flower classification task.

Table 2. Swin_transformer_model with Hyper parameter variation.

Algorithm: create_swin_transformer_model	Algorithm: create_swin_transformer_model
Input: - image_size: Tuple of integers representing the input image size (224, 224). - Output: - model: Swin Transformer model. Steps: 1. Initialize a sequential model. 2. Add a Swin Transformer layer to the model with the following parameters: - image_size: 224x224. - patch size: 4 # Adjust patch size based on needs. - num_heads: 8 - num_layers: 6 - mlp_dim: 256 - channels: 64 - num_classes: Number of output classes. - head_size: 16 . - window size: 7 . - shift_size: 0 . - dropout_rate: 0.1 3. Return the created model. Usage: - Set image_size = (224, 224) - Set num_classes = 104 to obtain the Swin Transformer model .	Input: - image_size: Tuple of integers representing the input image size (512, 512). Output: - model: Swin Transformer model. Step : 1. Initialize a sequential model. model = Sequential() Step 2: Add a Swin Transformer layer to the model with modified parameters. model.add(SwinTransformer(image_size=image_size, patch size=4, num_heads=12, num_layers=8, mlp_dim=256, channels=64, num_classes=num_classes, head_size=32, window size=7, shift_size=0, dropout_rate=0. Step 3: Return the created model. return model Usage: Set image_size = (512, 512) Set num_classes = 104 to obtain the Swin Transformer model.

8 Result and Discussion

Table 3: Comparison of results

Metric	:Value	Metric	:Value
Best Epoch	: 21	Best Epoch	: 18
Best Val Loss	: 0.19509	Best Val Loss	: 0.314
Final Epoch	: 21	Final Epoch	: 21
F1 Score	:0.99793	F1 Score	: 92.85%
Loss	: 0.00728	Loss	: 0.314
Learning Rate	: 1.56e-05	Learning Rate	: 1e-05
Val F1 Score	: 0.96603	Val F1 Score	: 92.85%
Val Loss	: 0.19509	Val Loss	: 0.314

Table 3 compares the models training results .The First model outperforms the second with lower validation loss (0.19509 vs. 0.314), higher F1-score (99.79% vs.92.85%), and better generalization (val F1 96.60% vs. 9285%).It also benefits from a slightly higher learning rate (1.56e-05 vs 1e-05), contributing to better optimization .overall the swin transformer model with configuration 1 (Random search) is preferred for deployment due to its superior accuracy, stability, and efficiency.

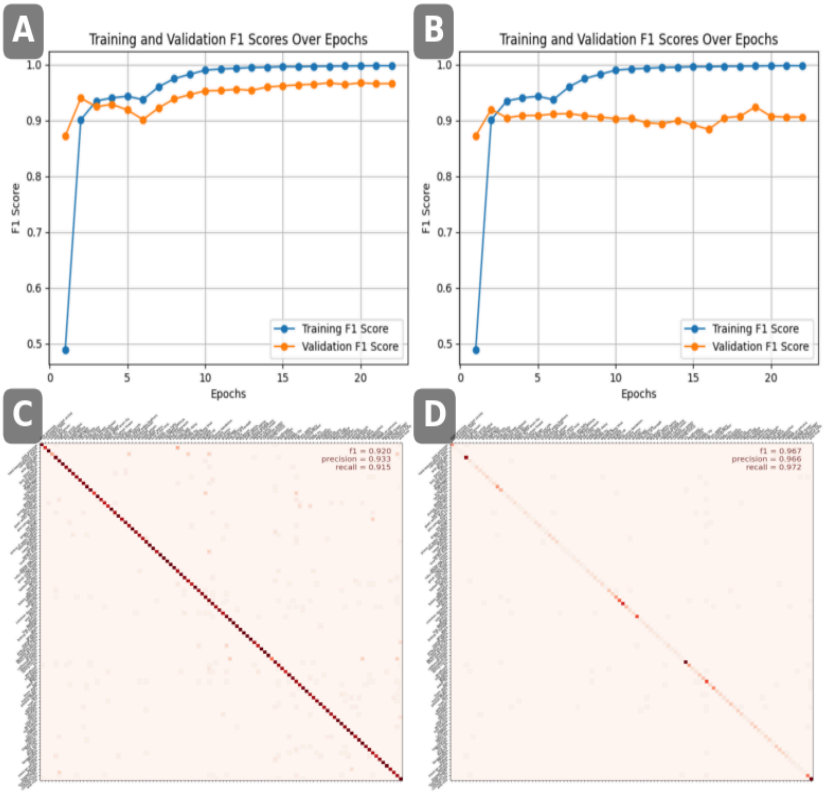

Fig 4. (A) shows Training and validation score using Random Search (B) score using Bayesian Optimization ,(C) Confusion matrix with Swin_transformer_model: Bayesian Optimization (D) Confusion matrix with Swin_transformer_model: Random Search.

Table 4: Performance matrix

Model	F-1 score	Precision	Recall
Swin_transformer_model: Random Search	0.9660	.966	.972
Swin_transformer_model: Bayesian Optimization	.9285	.933	.915

With a low loss of 0.00728 in figure 4, the model reaches an exceptionally high F1 score of 96.60% in the best epoch (Epoch 21). The rate of learning is quite modest (2e-05). At the best epoch (Epoch 18), the model in figure 4 gets a lower F1 score of 92.85% with a larger loss of 0.314. The rate of learning is significantly higher (1e-02).

9 Conclusion

The results showed that different optimization strategies performed differently, with the model that was tweaked using Random Search showing better performance metrics. For automated flower categorization systems to develop in the future, more research into Hyper parameters and dataset optimization is still necessary. By laying the foundation for future improvements in dataset duration and model efficiency, this research helps to develop more reliable and accurate floral categorization models.

10 Future Work

The future work can be Real-Time Deployment and Efficiency: Maintaining accuracy while taking into account inference speed and model efficiency can help in optimizing the model for real-time deployment. This might entail changing the architecture or using model compression techniques.

References

1. Narvekar, C., Rao, M.: Flower classification using CNN and transfer learning in CNN—Agriculture perspective. IEEE 3(12), 660–664 (2020). https://doi.org/10.1109/ICISS49785.2020.9316030
2. Jannat, F.-E., Willis, A. R.: Improving classification of remotely sensed images with the Swin Transformer. IEEE 4(7), 611–618 (2022). https://doi.org/10.1109/SoutheastCon48659.2022.9764016
3. Kaya, A., Keceli, A. S., Catal, C., Yalic, H. Y., Temucin, H., Tekinerdogan, B.: Analysis of transfer learning for deep neural network-based plant classification models. Comput. Electron. Agric. 2(5), 20–29 (2019). https://doi.org/10.1016/j.compag.2019.01.041
4. Thenmozhi, K., Reddy, U. S.: Crop pest classification based on deep convolutional neural network and transfer learning. Comput. Electron. Agric. 5(8), 104906 (2019). https://doi.org/10.1016/j.compag.2019.104906
5. Erkan, U., Toktas, A., Ustun, D.: Hyperparameter optimization of deep CNN classifier for plant species identification using artificial bee colony algorithm.

J. Ambient Intell. Humaniz. Comput. 6(9), 8827–8838 (2023). https://doi.org/10.1007/s12652-021-03631-w
6. Tiwari, V., Joshi, R. C., Dutta, M. K.: Deep neural network for multi-class classification of medicinal plant leaves. Expert Syst. 7(3), 13041 (2022). https://doi.org/10.1111/exsy.13041
7. Lu, J., Tan, L., Jiang, H.: Review on convolutional neural network (CNN) applied to plant leaf disease classification. Agriculture 8(4), 707 (2021). https://doi.org/10.3390/agriculture11080707
8. Naidu, G., Zuva, T., Sibanda, E. M.: A review of evaluation metrics in machine learning algorithms. Lecture Notes in Networks and Systems 9(6), 724 (2023). https://doi.org/10.1007/978-3-031-35314-7_2
9. Turner, R., Eriksson, D., McCourt, M., Kiili, J., Laaksonen, E., Xu, Z., Guyon, I.: Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the Black-Box Optimization Challenge 2020. PMLR 10(2), 3–26 (2021).
10. Kaggle.: TFRecords to PNG & CSV. Kaggle 11(5) (n.d.). Retrieved from https://www.kaggle.com/code/alenic/tfrecords-to-png-csv
11. Xie, J., Hua, J., Chen, S., Wu, P., Gao, P., Sun, D., Lyu, Z., Lyu, S., Xue, X., Lu, J.: HyperSFormer: A transformer-based end-to-end hyperspectral image classification method for crop classification. Remote Sens. 15(3), 3491 (2023). https://doi.org/10.3390/rs15143491