Proceedings of the 2nd International Conference on Software Engineering and Machine Learning
DOI: 10.54254/2755-2721/76/20240583

Weather forecast image classification based on KNN, CNN, and VGG19

Chunzhu Meng

The University of Hong Kong, Hong Kong, 999077, China

u3620225@connect.hku.hk

Abstract. Image classification is an important technology in the field of computer vision, with the main objective of categorizing input images into different classes. It is used in many areas, such as weather forecasting. The significance of weather forecast image classification spans several areas, including enhancing weather forecast accuracy, aiding agriculture, optimizing the energy sector, ensuring transportation safety, and informing urban planning and construction. In this study, the author investigates the precision and efficiency of three machine learning algorithms—Convolutional Neural Network (CNN), K-Nearest Neighbor (KNN), and Visual Geometry Group 19 (VGG19)—in classifying weather forecast images. The research meticulously elucidates the methodologies underlying each algorithm, outlining their distinct architectures, training processes, and feature extraction mechanisms. The research highlights that the self-built CNN model excels in accuracy and efficiency, while VGG19 achieves higher accuracy (93%) with longer training time. In contrast, the KNN model shows shorter training time (32.50 seconds) with slightly lower accuracy (80%).

Keywords: Weather Forecast, Image Classification, CNN, KNN, VGG19.

1. Introduction

Image classification techniques have wide applications in various fields, such as autonomous driving [1], security surveillance [2], medical imaging [3], smart homes [4], and more. As a crucial module within the broader field of image classification, weather forecast image classification significantly influences people's daily lives by affecting solar energy systems, sports events, and the functionality of various visual systems such as outdoor video surveillance and vehicle assistance systems. Unlike typical image classification tasks, weather forecast image classification is challenged by factors like lighting, reflection, scenery, and shadows, which intertwine to create a highly nonlinear categorization problem. To address these complexities, researchers have extensively explored the use of Convolutional Neural Networks (CNNs) and deep learning technologies [5-9], demonstrating that CNNs, with their end-to-end convolutional architecture, can simplify weather forecast image classification without the need for intricate feature engineering. Deep CNNs, in particular, have shown remarkable discriminative power across a range of image representation and classification tasks. Studies have incorporated various techniques, including integrating AlexNet and ResNet CNNs with multi-class Support Vector Machines (SVM) [6], leveraging pre-trained ImageNet-CNNs and weather-specific CNN networks [7], and combining Generative Adversarial Networks with transfer learning [8]. Additionally, models like InceptionV3, MobileNetV2, and ResNet50 [9] have been tailored for weather recognition, alongside

DOI: 10.54254/2755-2721/76/20240583

hybrid methods that combine Principal Component Analysis (PCA) with Linear Discriminant Analysis (LDA) [10]. Despite these advancements, previous models commonly suffer from issues of low accuracy and a lack of comparative analysis between models.

In this paper, the author performs a predictive task of classifying weather images into four types: cloud, rain, shine, and sunrise. The second section introduces the principles of the three machine learning algorithms used in this paper, including CNN, KNN, and VGG19 models. The third section discusses the selected dataset and data preprocessing methods mainly focused on data augmentation, including horizontal image flipping, image rotation, Gaussian blur, and brightness alteration. The fourth section presents model accuracy and efficiency evaluation metrics, followed by corresponding model results comparison and analysis. This comparative analysis sheds light on the trade-offs between accuracy and efficiency in weather image classification models, providing valuable insights for practical applications in meteorology.

2. Machine learning algorithms

In this section, three machine learning techniques, KNN, CNN, and VGG19, are discussed.

2.1. KNN

The K-Nearest Neighbor (KNN) algorithm is a key supervised learning method used for classification or regression tasks. It assigns class labels or predicted values to points in the feature space representing n-dimensional input vectors. Selecting the appropriate k value is crucial in the KNN algorithm (see Fig. 1). A smaller k leads to higher model complexity and lower training error, but weaker generalization ability; while a larger k reduces model complexity, increases training error, yet improves generalization to some extent. This paper employs cross-validation to iteratively determine the optimal k value, ultimately identifying 11 as the most suitable choice.

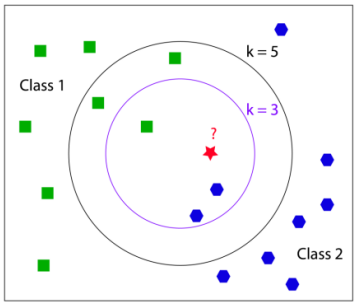

Figure 1. Basic schematic diagram of KNN [11].

There are two commonly used distance metrics in the KNN algorithm: Euclidean Distance and Manhattan Distance. The Euclidean Distance is used in this paper due to its simplicity and robust efficacy. For two n-dimensional vectors $a(x_{11}, x_{12}, \ldots, x_{1n})$ and $b(x_{21}, x_{22}, \ldots, x_{2n})$, the formula for calculating Euclidean Distance is:

$$d_{12} = \sqrt{\sum_{k=1}^{n} (x_{1k} - x_{2k})^2} \tag{1}$$

This method utilizes bicubic interpolation for image resizing, which retains finer details, providing smoother results than alternative methods. The calcHist function from OpenCV computes a two-dimensional histogram of the input image, and the normalized image histogram is flattened into a one-dimensional array.

2.2. CNN

CNNs, or Convolutional Neural Networks, are deep learning algorithms tailored for analyzing grid-structured data like images. Before entering the network, data in this study undergoes preprocessing steps including normalization and PCA. The journey starts with the Convolutional Layer,

DOI: 10.54254/2755-2721/76/20240583

where convolutional kernels process the input data with parameters like kernel size, stride, and padding. The convolutional layer output undergoes nonlinear processing using activation functions to enhance the model's expressive capability. Following is the pooling layer, where maximum pooling is applied in this study. It includes dividing the input feature map into non-overlapping rectangular regions (pooling windows), with the maximum value within each window serving as the output. The fully connected layer follows, connecting neurons across layers with learned weights. Results are then produced using the Softmax function.

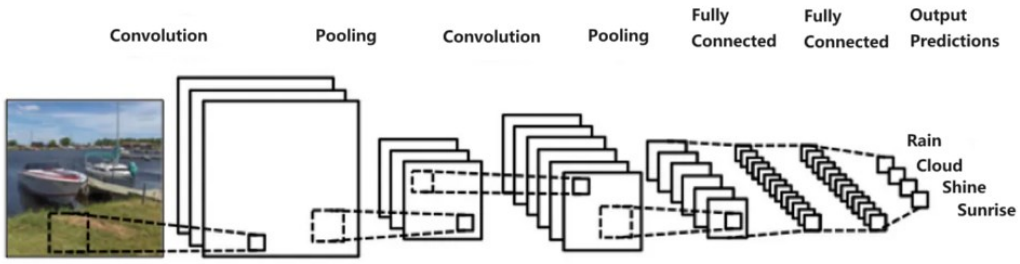

Figure 2. Basic schematic diagram of CNN.

In this method, the defined layers and operations are as follows (see Fig. 2): Convolutional layer conv0: Comprising 20 kernels of size 5 with ReLU activation. Max pooling layer pool0: Using a 2 x 2 pooling window with a stride of 2 x 2. Convolutional layer conv1 and max pooling layer pool1: Similar to conv0 and pool0, with 40 kernels of size 4 and a 2 x 2 pooling window with a stride of 2 x 2. Fully connected layer: Converting the flattened feature vector to a 400-feature-vector, using ReLU activation, and applying Dropout to prevent overfitting. The original labels are converted to one-hot encoding form, and the average cross-entropy loss is calculated for all samples after applying the Softmax activation function. The neural network parameters are updated via backpropagation using the Adam optimizer with a learning rate of 0.0001 to minimize the loss function.

The expression for the Softmax function is:

$$p_i = Softmax(z_i) = \frac{e^{z_i}}{\sum_{i=1}^{C} e^{z_i}} \tag{2}$$

Where i represents the index of the output node, p_i is the probability value for output category i. The expression for the cross-entropy loss function is:

$$L = -\sum_{i=1}^{C} y_i \, log(p_i) \tag{3}$$

2.3. VGG19

VGG19 [12] is a deep convolutional neural network model. It uses a series of convolutional layers to extract image features. These convolutional layers typically use small 3 x 3 convolutional kernels with a stride of 1 to maintain spatial resolution. The non-linear fitting capability is increased by stacking multiple convolutional layers. Following the convolutional layers, max-pooling layers are used to reduce the spatial dimensions of feature maps, decrease the number of parameters, and extract more prominent features. Additionally, the parameter sharing mechanism of the convolutional layers makes the model training more efficient and enables it to handle input images of different sizes.

In this method, all images are standardized to 256 x 256 dimensions while retaining the RGB color scheme. A convolutional neural network, designed to mimic the VGG19 model architecture, is utilized in this study. The network comprises 5 convolution blocks (see Fig. 3): 2 x 64, 2 x 128, 4 x 256, 4 x 512, and 4 x 512, each followed by a 2 x 2 pooling layer. Each convolution kernel size is set to 3 x 3. Subsequent to the convolution blocks, the model is flattened and connected to a 1024-depth dense layer, followed by a dropout of 0.3 to prevent overfitting. This is succeeded by a 64-depth dense layer, and finally a 4-depth dense output layer. The ReLU activation function is employed for relevant layers,

DOI: 10.54254/2755-2721/76/20240583

except for the output layer which uses Softmax activation since it is a multi-classification problem. The Adam optimizer is applied to minimize the categorical cross-entropy loss function. Furthermore, the learning rate can be automatically adjusted during training to improve efficacy.

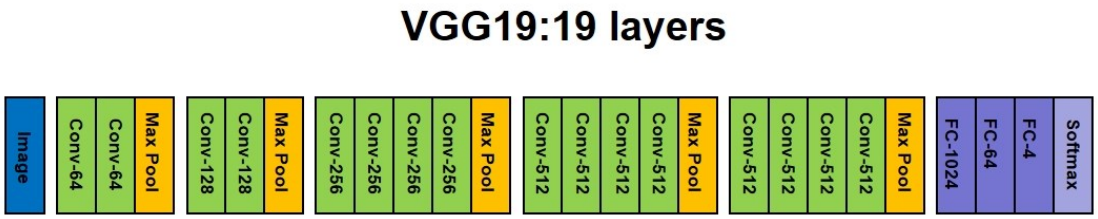

Figure 3. Visualization of the VGG19 model architecture.

3. Data set and data processing

3.1. Data set: weather forecast data set

This dataset comprises images from four categories: "Cloud", "Rain", "Shine", and "Sunrise" (see Fig. 4). There are a total of 1123 images, with the following distribution per category:

(a) "Cloud": 300 (b) "Rain": 214 (c) Shine": 252 (d) "Sunrise": 357

Figure 4. Initial dataset structure.

This dataset is utilized for image classification tasks and encompasses a variety of images depicting different weather conditions and sunrise scenes. It is divided into training, validation, and test sets, and the images undergo preprocessing and standardization. Additionally, data augmentation techniques can be employed to expand the dataset and enhance the model's generalization capabilities.

3.2. Data processing

Deep learning neural network models demand a vast number of parameters, often reaching into the millions or more. This highlights the need for ample high-quality data to effectively operate and train the model. In the context of this study, the weather forecast dataset consists of only 200-300 images per category, indicating a scarcity of samples. To bolster the model's generalization capacity, four data augmentation techniques are utilized to expand the dataset for training and testing (see Fig. 5).

Horizontal image flipping: Horizontal image flipping is a common data augmentation technique that generates new training samples by mirroring images from left to right. This technique increases dataset diversity and enhances model generalization without altering the image category labels.

Gaussian blur processing: Gaussian blur, a common data augmentation technique, simulates slight image blurring to help deep learning models adjust to real-world noise. It involves applying a Gaussian filter around each pixel in the image. This process aids models in handling noise and improving generalization.

Brightness alteration: Brightness alteration is a common technique in data augmentation. By adjusting image brightness, a range of images with different brightness levels is created, diversifying the training data and enabling the model to adapt effectively to varying lighting conditions. In this article, the brightness is increased by 50%.

DOI: 10.54254/2755-2721/76/20240583

Image rotation: Rotating images can generate images with different perspectives, increasing data diversity and helping to enhance the generalization and robustness of deep learning models. In this article, the image is rotated 90 degrees clockwise.

Figure 5. Data augmentation image example.

The expanded dataset contains 5611 samples, distributed across weather categories as follows: "Cloud": 1500 images; "Rain": 1070 images; "Shine": 1256 images; "Sunrise": 1785 images. These extra samples, created through data augmentation, enrich the original dataset, leading to a more balanced distribution across different weather conditions. Such a balanced distribution helps the model learn features and patterns under various weather conditions more comprehensively, consequently improving the performance and robustness of the weather image classification model.

4. Performance analysis

There are four different ways to define correct and incorrect results: True Negative (TN): Case that was negative and predicted to be negative; True Positive (TP): Case that was positive and predicted to be positive; False Negative (FN): Case that was positive but predicted to be negative; False Positive (FP): Case that was negative but predicted to be positive.

The formulas for accuracy and recall are given as follows:

$$\text{Classfication accuracy} = \frac{TP+TN}{TP+FP+FN+TN}, \text{Recall} = \frac{TP}{TP+FN} \tag{4}$$

The formulas for precision and F1-score are given as follows:

$$\text{Precision} = \frac{TP}{TP+FP}, \text{F-measure} = \frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}} \tag{5}$$

4.1. KNN result

In the KNN experiment, the dataset was divided into training and testing sets in an 80% to 20% ratio, with the training set containing 4488 images and the testing set containing 1123 images. Based on the performance of the KNN algorithm on the testing set, the following results were obtained: The number of correctly predicted samples is 898. This calculation yields an accuracy of 0.80, or 80%. The algorithm evaluation is shown in Table 1:

Table 1. KNN algorithm evaluation.

	Precision	**Recall**	**F1-score**	**Support**
0:	0.83	0.57	0.68	301
1:	0.65	0.77	0.70	213
2:	0.86	0.87	0.87	259
3:	0.83	0.96	0.89	350
Accuracy	0.80			1123

DOI: 10.54254/2755-2721/76/20240583

Table 1. (continued).

Macro avg.	0.79	0.79	0.78	1123
Weighted avg.	0.80	0.80	0.79	1123

The Confusion Matrix, ROC curve, Precision-recall curve, and PCA projection of the dataset for the KNN algorithm are shown below:

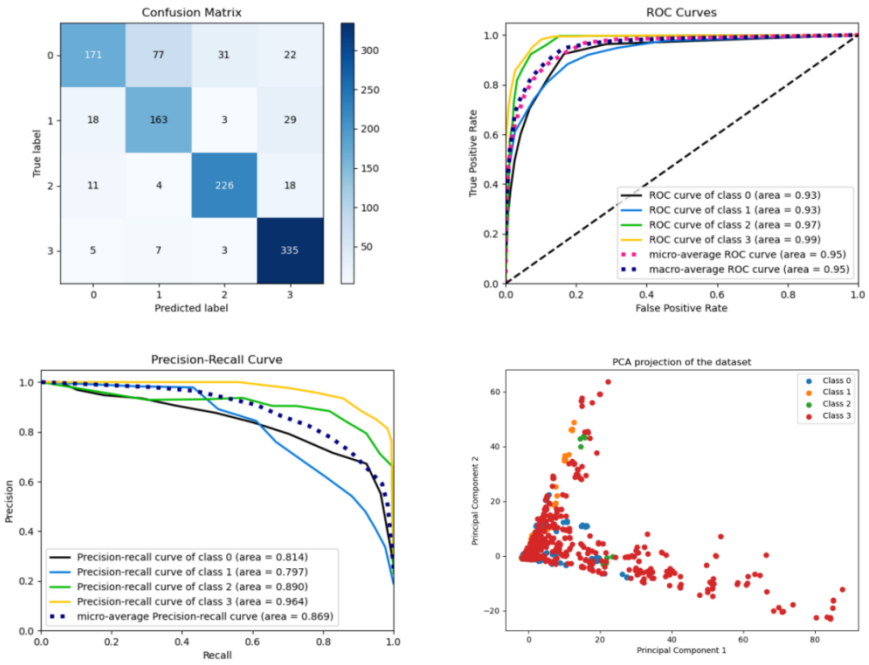

Figure 6. The Confusion Matrix, ROC curve, Precision-recall curve, and PCA projection of the dataset for the KNN algorithm. (a) AUC-ROC ranges from 0 to 1, with higher values indicating better discrimination between positive and negative samples at different thresholds. (b) AUC-PR ranges from 0 to 1, with higher values indicating better performance. (c) PCA 2D maps data points onto a new two-dimensional space using two principal component values, derived from variables with the highest variance.

From Fig. 6, it can be seen that the AUC-ROC values for each class are all above 0.9, with a micro-average AUC-ROC considering the classification results of multiple classes of 0.95. The micro-average AUC-PR is 0.869. The four classes do not exhibit distinct distribution patterns in the PCA 2D plot, and the data labeled "sunrise" show significant structural differences.

4.2. CNN result

In the CNN experiment, the dataset was also divided into training and testing sets in an 80% to 20% ratio, with the training set containing 4476 images and the testing set containing 1120 images. Based on the performance of the CNN algorithm on the testing set, the following results were obtained: The number of correctly predicted samples is 1086. This calculation yields an accuracy of 0.97, or 97%. The algorithm evaluation is seen in Table 2:

Table 2. CNN algorithm evaluation.

	Precision	**Recall**	**F1-score**	**Support**
0:	0.94	0.97	0.95	293
1:	0.97	0.97	0.97	239

DOI: 10.54254/2755-2721/76/20240583

Table 2. (continued).

2:	0.96	0.93	0.94	238
3:	0.99	0.99	0.99	350
Accuracy	0.97			1120
Macro avg.	0.96	0.96	0.96	1120
Weighted avg.	0.97	0.97	0.97	1120

The Confusion Matrix, ROC curve, Precision-recall curve, and PCA projection of the dataset for the CNN algorithm are shown below:

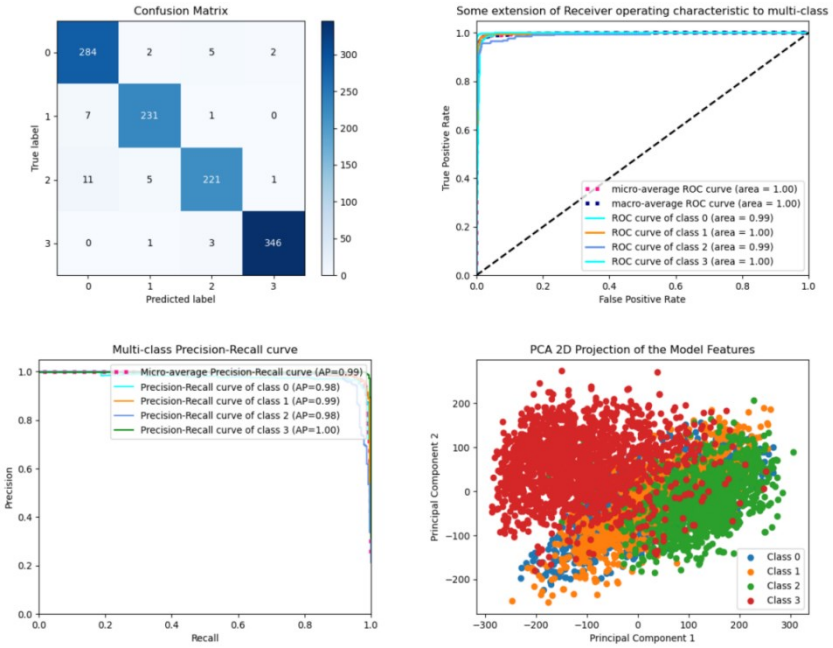

Figure 7. The Confusion Matrix, ROC curve, Precision-recall curve, and PCA projection of the dataset for the CNN algorithm.

From Fig. 7, it can be observed that the AUC-ROC values for each category are all 0.99 or higher, with a micro-average AUC-ROC considering the classification results of multiple categories of 1. The micro-average AUC-PR is 0.99. The four categories exhibit significantly different distribution patterns in the PCA 2D plot, indicating a good clustering effect.

4.3. VGG19 result

In the VGG19 experiment, the dataset was also divided into training and testing sets in an 80% to 20% ratio, with the training set containing 4491 images and the testing set containing 1120 images. Based on the performance of the VGG19 algorithm on the testing set, the following results were obtained: The number of correctly predicted samples is 1037. This calculation yields an accuracy of 0.926, or 93%. The algorithm evaluation is shown in Table 3:

Table 3. VGG19 algorithm evaluation.

	Precision	**Recall**	**F1-score**	**Support**
0:	0.85	0.94	0.89	300
1:	0.92	0.86	0.89	215
2:	0.97	0.86	0.91	250

DOI: 10.54254/2755-2721/76/20240583

Table 3. (continued).

3:	0.97	1.00	0.98	355
Accuracy	0.93			1120
Macro avg.	0.93	0.92	0.92	1120
Weighted avg.	0.93	0.93	0.93	1120

The Confusion Matrix, ROC curve, Precision-recall curve, and PCA projection of the dataset for the VGG19 algorithm are shown below:

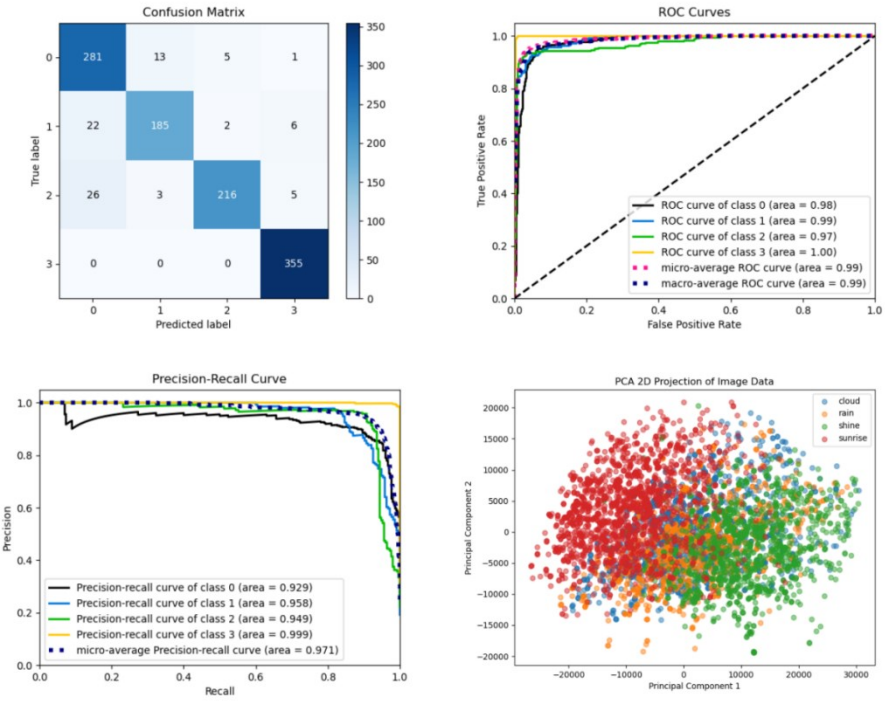

Figure 8. The Confusion Matrix, ROC curve, Precision-recall curve, and PCA projection of the dataset for the VGG19 algorithm.

From Fig. 8, it can be observed that the AUC-ROC values for each category are all 0.97 or higher, with a micro-average AUC-ROC considering the classification results of multiple categories of 0.99. The micro-average AUC-PR is 0.97. The four categories exhibit significantly different distribution patterns in the PCA 2D plot, indicating a good clustering effect.

4.4. Comparative analysis

According to Table 4, the self-built CNN model attains the highest accuracy of 97%, followed by the VGG19 model at 93%. The KNN model achieves the lowest accuracy of 80%. In terms of training time, the KNN model is the quickest, finishing in 32.50 seconds. In contrast, the self-built CNN model requires 147.39 seconds for training, while the VGG19 model takes the longest time at 1587.58 seconds.

Table 4. Algorithm Comparison.

	KNN	**CNN**	**VGG19**
Accuracy	80%	97%	93%
Execution Time(sec)	32.50	147.39	1587.58

DOI: 10.54254/2755-2721/76/20240583

5. Conclusion

In this paper, a performance comparison of different classifier models for classifying images over the weather forecast dataset is conducted. The models considered for comparison are KNN, CNN, and VGG19. Accuracy and metrics such as recall, precision, and F1 score are taken into account for the comparison. In summary, the self-built CNN model excels in accuracy and training efficiency. VGG19 achieves high accuracy (93%) with longer training time (1587.58s). In contrast, the KNN model has shorter training time (32.50s) with slightly lower accuracy (80%), while the self-built CNN model outperforms in both accuracy (97%) and training time (147.39s). However, it does not necessarily mean that the self-built CNN model is the optimal choice. Other factors, such as model complexity, storage space, and computational cost, need to be considered.

The current model achieves a maximum accuracy of 97%. To further enhance the accuracy of weather forecasting, future studies can experiment with more complex models, increase the training dataset size, adjust hyperparameters, and ensemble models like KNN and VGG19. Improving image preprocessing can boost data quality. Model compression can reduce training time. Also, the models developed for weather forecast image classification can be applied to other fields such as medical imaging and industrial quality inspection.

References

[1] Moura, T., Valente, A., Sousa, A. and Filipe, V. (2014). Traffic Sign Recognition for Autonomous Driving Robot. In 2014 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), Espinho, Portugal, 303-308.

[2] Hulaj, A., Shehu, A. and Bajrami, X. (2017). Support Vector Machine for the Classification of Images Captured by WMSN. In 2017 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO), Prague, Czech Republic, 283-287.

[3] Cai, L., Gao, J. and Zhao, D. (2020). A review of the application of deep learning in medical image classification and segmentation. Annals of Translational Medicine, 8(11), 713.

[4] Hassan, A., Liu, F., Wang, F., et al. (2021). Secure image classification with deep neural networks for IoT applications. Journal of Ambient Intelligence and Humanized Computing, 12, 8319-8337.

[5] Cao, Y. and Yang, H. (2022). Weather Prediction using Cloud's Images. In 2022 International Conference on Big Data, Information and Computer Network (BDICN), Sanya, China: IEEE, 820-823.

[6] An, J., Chen, Y. and Shin, H. (2018). Weather Classification using Convolutional Neural Networks. In 2018 International SoC Design Conference (ISOCC), Daegu, Korea (South): IEEE, 245-246.

[7] Elhoseiny, M., Huang, S. and Elgammal, A. (2015). Weather classification with deep convolutional neural networks. In 2015 IEEE International Conference on Image Processing (ICIP), Quebec City, QC, Canada: IEEE, 3349-3353.

[8] Zou, Y., Wu, J., Chen, W. and Liu, Y. (2022). Weather image classification based on generative adversarial network and transfer learning. In Proceedings Volume 12165, International Conference on Intelligent Traffic Systems and Smart City (ITSSC 2021), paper 121651S.

[9] Młodzianowski, P. (2022). Weather Classification with Transfer Learning - InceptionV3, MobileNetV2 and ResNet50. In C. Biele, J. Kacprzyk, W. Kopeć, J.W. Owsiński, A. Romanowski, M. Sikorski (Eds.), Digital Interaction and Machine Intelligence. In Proceedings of MIDI 2021 (Lecture Notes in Networks and Systems, Vol. 440). Springer, Cham.

[10] Hapsari, Y. and Syamsuryadi. (2019). Weather Classification Based on Hybrid Cloud Image Using Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA). In 2nd Forum in Research, Science, and Technology, October 30-31, 2018, Horizon Ultima Hotel, Palembang, Indonesia (Vol. 1167). Journal of Physics: Conference Series. IOP Publishing Ltd.

DOI: 10.54254/2755-2721/76/20240583

[11] Asher117. (2019). Image reference: CSDN K-Nearest Neighbors Algorithm Principles. CSDN Blog. Retrieved on March 20, 2024, from https://blog.csdn.net/Asher117/article/details/96115458.
[12] Visual Geometry Group (VGG). (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. Retrieved from https://arxiv.org/abs/1409.1556.